НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

(повне найменування інституту, факультету)

Автоматизованих систем обробки інформації і управління

(повна назва кафедри)

«До захисту допущено»
В.о. завідувача кафедри
_____ *О.А.Павлов*
(підпис) (ініціали, прізвище)
“ ” 2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки ***6.050103 «Програмна інженерія»***

на тему *Комплекс задач з автоматизації варіативних опитувань*

Виконав: студент IV курсу, групи *ІП-51 Павлюк Вадим Русланович* _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *ст.викл. Ковтунець О.В.* _____
посада,науковий ступінь,вчене звання,прізвище,ініціали (підпис)

Консультант з графічної документації *доцент, к.т.н Ліщук К.І.* _____
посада,науковий ступінь,вчене звання,прізвище,ініціали (підпис)

Рецензент:

ст.викл.каф. ОТ Саверченко В.Г.
посада,науковий ступінь,вчене звання,прізвище,ініціали (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) *Інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації і управління*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки (програма професійного спрямування) – *6.050103 «Програмна інженерія» (Програмне забезпечення систем)*

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *О.А. Павлов*
(підпис) (ініціали, прізвище)
“___”_____2019 р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТУ

Павлюку Вадиму Руслановичу
(прізвище, ім`я, по батькові)

1. Тема проекту «*Комплекс задач з автоматизації варіативних опитувань*»

керівник проекту *Ковтунець Олесь Володимирович, старший викладач*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від “23” квітня 2019 р. №1181-с

2. Термін подання студентом проекту «03» *червня 2019 року*

3. Вихідні дані до проекту
Технічне завдання

4. Зміст пояснювальної записки
1) Аналіз вимог до програмного забезпечення: основні визначення та терміни, опис предметного середовища, огляд існуючих технічних рішень та відомих програмних продуктів, розробка функціональних та нефункціональних вимог
2) Моделювання та конструювання програмного забезпечення: моделювання та аналіз програмного забезпечення, засоби розробки, технічні рішення, архітектура програмного забезпечення
3) Аналіз якості та тестування програмного забезпечення

4) Розгортання та впровадження програмного Забезпечення, керівництво Користувача

5. Перелік графічного матеріалу

1) Схема структурна варіантів використань

2) Схема структурна класів програмного забезпечення

3) Креслення вигляду екранних форм

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «12» *березня 2018 року*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	*Вивчення рекомендованої літератури*		
2.	*Аналіз існуючих методів розв'язання задачі*		
3.	*Постановка та формалізація задачі*		
4.	*Аналіз вимог до програмного забезпечення*		
5.	*Алгоритмізація задачі*		
6.	*Моделювання програмного забезпечення*		
7.	*Обґрунтування використовуваних технічних засобів*		
8.	*Розробка архітектури програмного забезпечення*		
9.	*Розробка програмного забезпечення*		
10.	*Налагодження програми*		
11.	*Виконання графічних документів*		
12.	*Оформлення пояснювальної записки*		
13.	*Подання ДП на попередній захист*	*28.05.2019*	
14.	*Подання ДП рецензенту*		
15.	*Подання ДП на основний захист*	*08.06.2019*	

Студент _____ Павлюк В.Р.
(підпис)

Керівник проекту _____ Ковтунець О.В.
(підпис)

Пояснювальна записка
до дипломного проекту

на тему: Комплекс задач з автоматизації варіативних опитувань _____
_____
_____

Київ – 2019 року

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЕКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проект	2	
2	А4	КПІ.ІП-XXXX. XXXXXX.02.81	<Тема ДП>.		
			Пояснювальна записка		
3	А4	КПІ.ІП-XXXX. XXXXXX.03.81	<Тема ДП>.		
			Технічне завдання		
4	А4	КПІ.ІП-XXXX. XXXXXX.04.51	<Тема ДП>.		
			Програма та		
			методика тестування		

АНОТАЦІЯ

Пояснювальна записка дипломного проекту складається з чотирьох розділів, містить 22 таблиці та 7 джерел – загалом 59 сторінок.

Об'єкт дослідження: електронні варіативні опитування, що реалізовані на основі динамічних веб-форм із логічно пов'язаними полями.

Мета дипломного проекту: створити програмне забезпечення для побудови гнучких варіативних форм із можливістю логічної взаємодії елементів опитування.

У першому розділі було здійснено аналіз існуючих програмних продуктів, описано функціональні та не функціональні вимоги, поставлено комплекс задач.

У другому розділі описано моделювання та конструювання программного забезпечення, проведено аналіз безпеки даних.

У третьому розділі описано аналіз якості та тестування программного забезпечення.

У четвертому розділі описано процесс розгортання та впровадження программного забезпечення, і також наведено інструкцію користувача.

КЛЮЧОВІ СЛОВА: ВАРІАТИВНІ ОПИТУВАННЯ, РОЗГАЛУЖЕНІ ОПИТУВАННЯ, ДИНАМІЧНІ ФОРМИ

ABSTRACT

The explanatory note of the diploma project consists of four sections, contains 22 tables and 7 sources - a total of 59 pages.

Object of research: electronic variational surveys implemented on the basis of dynamic web forms with logically related fields.

The purpose of the diploma project: to create software for the construction of flexible variational forms with the possibility of logical interaction of the survey elements.

In the first section, an analysis of existing software products was performed, functional and non-functional requirements were described, a set of tasks was set.

The second section describes the simulation and design of the software, conducted a data security analysis. The third section describes the quality analysis and software testing. The fourth section describes how to deploy and implement the software, and also the user's manual.

KEY WORDS: VARIATIVE EXAMINATIONS, DETERMINED EXAMINATIONS, DYNAMIC FORMS

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

SPA - Single page application - одно-сторінковий додаток, який працює на принципі попереднього завантаження одного файлу вихідного коду для подальшої взаємодії додатку.

URL - Uniform Resource Locator - стандартизована адреса певного ресурсу

Бандл – результуючий файл, після виконання мініфікації коду.

ВСТУП

Розвиток інформаційних технологій суттєво змінив життя сучасної людини, і продовжує робити це сьогодні. Зміни полягають у миттєвому доступі з майже будь-якої точки нашої планети до величезних об’ємів інформації у різному вигляді, від повідомлень у соціальних мережах до потокового-відео запуску ракет, від звичайних книг до коефіцієнтів котирування акцій на фондових ринках у реальному часі. Звичайно, що все ще можливо насамперед завдяки появі інтернету. Його історія досить неоднозначна, і він міг бути відмінним від того, що є сьогодні. Було багато ідей та концепцій об’єднання окремих вузлів у мережі, але успішною стала саме децентралізована система, де кожен учасник може, як користуватися інформацією, так і створювати її. Саме призведе до буму блогів, соціальних мереж та програм обміну повідомленнями.

Взагалі обмін інформацією історично є одним з найважливіших факторів еволюції людства. Первісні люди вигадали велику кількість способів цього обміну. Окремі з них мали неймовірний вплив. Саме початковий обмін звуками, потім переросте у налагоджену систему - мову. Тепер люди могли вільно поширювати свої думки. Цікавим аспектом цього процесу є опитування. Коли один учасник надає конкретну, здебільшого однозначну, відповідь на запит іншого.

Окремою категорією є масові опитування. Масові опитування мають велику соціальну роль. Вони можуть бути джерелом окремої, нової інформації, до цього не існувавшої, а отриманої саме в ході опитування. Реалізація саме цього класу опитувань особливо виправдана з використанням сучасних інформаційних технологій.

Самі опитування можуть значно відрізнятися по своїй внутрішній структурі. Кожне опитування складається з різних питань. Питання можуть бути з відкритою відповіддю, або з конкретними варіантами. Самі питання можуть залежати одне від одного, формуючи складну розгалужену систему, де результат попереднього кроку може змінити наступні. Такі опитування мають

клас варіативних, тобто варіативних. Саме цьому класу приділено найменше уваги на сучасному просторі інформаційних технологій.

І це незважаючи на те, наскільки вони поширені в окремих областях людської діяльності. Наприклад, вони є невід'ємною частиною величезної сфери страхування. Навіть, базові процеси в якій, потребують великої кількості уточнюючої інформації. Проблема відсутності ефективного інструменту створення варіативних опитувань може бути навіть причиною сповільненого процесу диджіталізації окремого ринку. Адже створення локальних рішень, вирішуючи локальні проблеми, окремо взятими компаніями може бути малоефективним у майбутньому, в процесі зміни вимог.

Інший ж підхід - створення автоматизованої системи варіативних опитувань, яка б, за допомогою своєї гнучкості, вирішувала проблеми можливої зміни вимог у майбутньому. Та проблему створення локальних рішень в цілому.

Саме тому метою дипломної роботи є комплекс задач з автоматизації варіативних опитувань, який є настільки необхідним у сучасних обставинах.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Опитування, як метод отримання соціологічної інформації щодо предмету опитування, по своїй структурі та формі та змісту досить різноманітні. Ті чи інші категорії краще підходять до відповідних їм задачам. Класифікація опитувань відбувається за якимось параметром. Широко використовуються поділ на категорії відносно методу опитування та відносно часу, впродовж якого відбувається опитування.

Серед методів опитування виділяють інтерв’ювання та анкетування. Інтерв’ювання полягає у безпосередньому контакті інтерв'юера (особи яка задає конкретні запитання) з респондентом (особа, яка надає відповіді на запитання інтерв'юера). Цей метод є основоположним в соціології. [1]

Він використовується у багатьох сферах людського життя. Від простої комунікації людей під час надання послуг, наприклад медичних, при уточнені деталей становища хворого, до повноцінного збору цілісної інформації у журналістиці, де інтерв’ювання є окремим жанром. У варіанті використання його у журналістиці, інтерв'юером являється журналіст, який використовує метод як спосіб отримання інформації, або опублікування роботи у вигляді самого інтерв'ю. Метод широко популярний у різних масових медіа: телевізійних програмах, радіо, в пресі. Тут також метод може представлятися у різних форматах: новинах, окремих програмах за участі відомих людей, дослідженнях та ін. Окремої уваги заслуговує інтерв’ювання у вигляді співбесіди. Найчастіше співбесіди в контексті прийому працівника на роботу. Інтереси опитувача може представляти інша довірена особа. У цьому вигляді інтерв’ювання несе змішані цілі. Це і метод отримання інформації, і дослідження як перевірка відповідності заявлених навиків та, як визначення можливої ефективності роботи респодента в межах необхідних задач. Цікавим та не широко відомим є наративне інтерв’ювання, яке характеризується

мінімальним втручанням інтерв'юєра, задля отримання максимальної кількості біографічної інформації від респодента. Суть полягає у створенні відповідних умов оповідання, як конкретних епізодів, на які вказує інтерв'юєр, або загального оповідання біографії з більшим освітленням окремих моментів, або заданням напряму інтерв'юєром.

Щодо іншого методу опитувань - анкетування, - основною відмінністю є наявність анкет - бланків з питаннями, які надаються респодентам для заповнюватися. Метод часто застосовується в маркетингових дослідженнях, для отримання інформації від можливих споживачів, щодо їх вподобань та відгуків щодо існуючої або майбутньої продукції. Якщо інтерв'ювання спрямоване на окремого респодента, то про анкетування можна сказати, що більшу цінність воно становить саме у опитуваннях багатьох респодентів. Взагалі анкетування можна класифікувати по декількох характеристикам.

За охопленням групи респодентів:

- суцільне (спрямоване на всю досліджувану групу);
- вибіркове (в процесі анкетування беруть участь лише деякі представники досліджуваної групи - вибірка, репрезентативність, якої встановлюється використовуючи статистику).

За кількістю респодентів:

- групове (в процесі анкетування одночасно бере участь декілька респодентів);
- індивідуальне (опитування респодентів відбувається окремо).

За відкритістю респодента:

- легальне (із зазначенням особи респондента);
- анонімне (без зазначення особи респондента).

Також опитування можна класифікувати за часом, необхідним для проведення опитування (короткочасні та тривалі опитування), де межі класифікації можуть відрізнятися залежно від сторонніх факторів. За форматом отримання відповідей (усні, письмові, електронні).

1.2 Опис процесу діяльності

Процес створення опитувань доволі складний процес, який можна розділити на декілька окремих етапів за виконанням головної ролі тим чи іншим учасником опитувань. [2]

Серед учасників можна виокремити сторону-замовника, який потребує отримання нової важливо для нього інформації, або перевірити власні гіпотези щодо якоїсь цільової аудиторії. Іншим учасником є експертна група, під керівництвом якої, потреба замовника зможе бути задоволеною у результаті відповідей на розроблені цією групою анкети опитування. Найчастіше представниками цієї групи є соціологи - фахівці, які проводять дослідження різних процесів у суспільстві.

Процес створення анкети в свою чергу поділяється на різні етапи.

Етапи створення анкет опитування:

- збір вимог, які має задовольняти опитування;
- трансформація вимог у елементарні складові опитувань - запитання (саме через серію запитань, опитування дають змогу отримати необхідні для дослідження дані);
- корегування питань відповідно до необхідних характеристик (найгловніші з них детермінованість та зрозумілість для респодента, який представлятиме цільову аудиторію дослідження), ці питання і будуть основними;
- додавання до основних питань, питання для збільшення довіри та пробудження зацікавленості на початку анкети, та питання на отримання демографічних даних у кінці опитування;
- створення запитань - фільтрів, для перевірки належності респодента до цільової аудиторії, якщо це не перевірено перед опитуванням;
- складання усіх питань разом у анкету, де спочатку йдуть фільтраційні питання, далі основні, які закінчуються демографічними;
- також можливе добавлення інструкції по заповненню анкети у верхній

частині.

Окремої уваги заслуговує етап корегування питань відповідно до базових характеристик (взагалі характеристики можуть відрізнятися залежно від задач опитування). [3]

До базової характеристики питань для опитувань належать:

- зрозумілість для респодентів (необхідно уникати спеціальних термінів, якщо немає їх окремого пояснення);
- однозначність (питання мають бути з можливістю надання конкретної відповіді);
- питання мають бути лаконічними;
- питання повинні бути без додатково підтексту, не містити відповідей у самому тілі запитання;
- не бути риторичними;
- не задівати особистих (інтимних) тем, бесіда на яку може здатися неприємною для респондентів;
- не містити навіювальних характер, коли у ході анкетування відбувається психологічний вплив на респондента;
- слід уникати абстрактних питань.

Далі, після отримання моделі анкети процес переходить до групи асистентів, які займаються виготовленням анкет та їх логістикою до респодентів. Респоденти надають відповіді, заповнюючи анкету, передають результати назад асистентам, вони в свою чергу передають зібрані анкети на анліз до групи соціологів.

Равершувальний етап - аналіз результатів опитування, підбиття висновків щодо замовленого дослідження або гіпотези та передача результатів замовнику.

Якщо спростити процес до головних дійових осіб, виключивши етап замовлення, та представивши експертну соціологом, групу асистентів - асистентом, отримаємо формалізовану схему зображену на рисунку 1.1

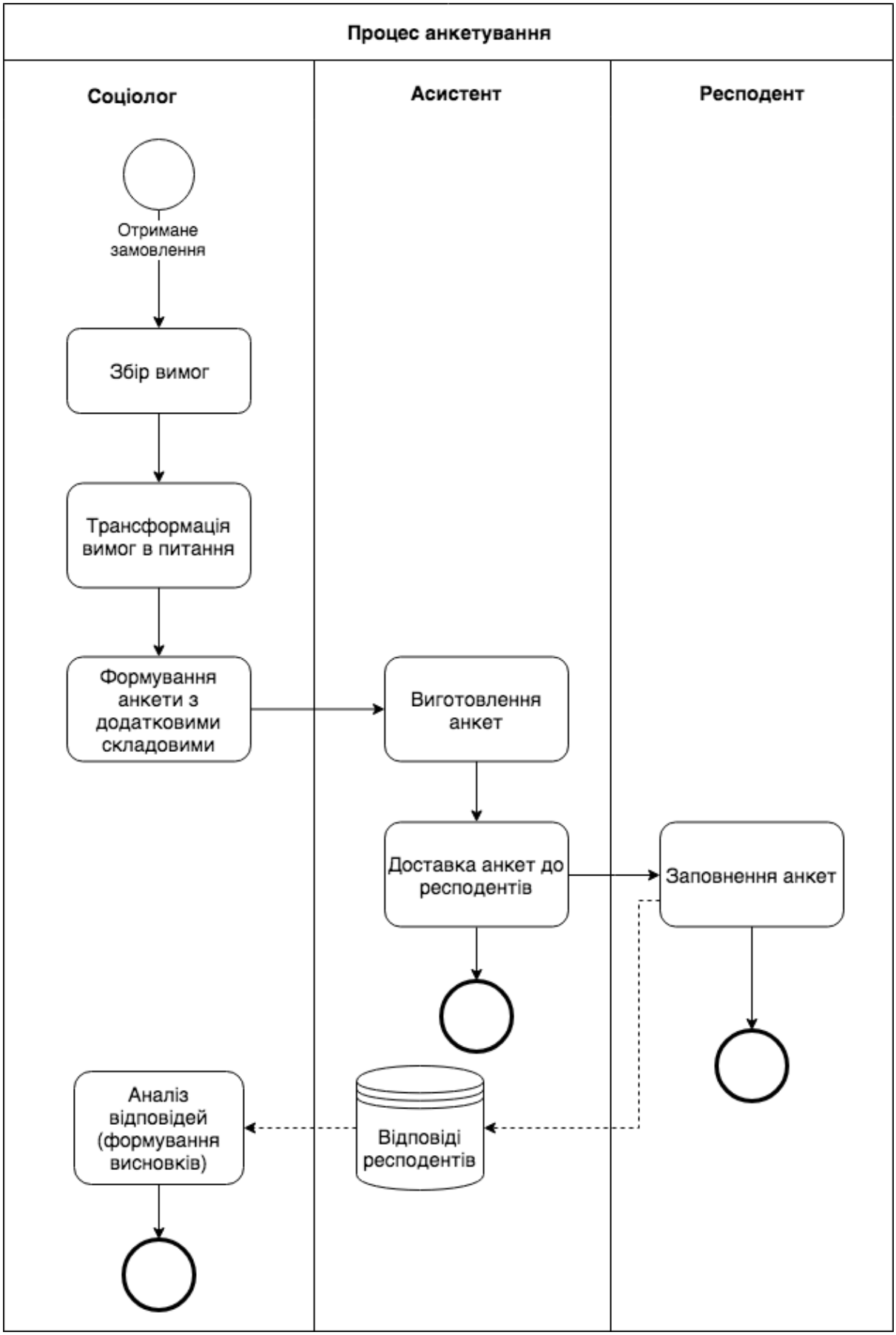


Рисунок 1.1 – Схема діяльності учасників опитування

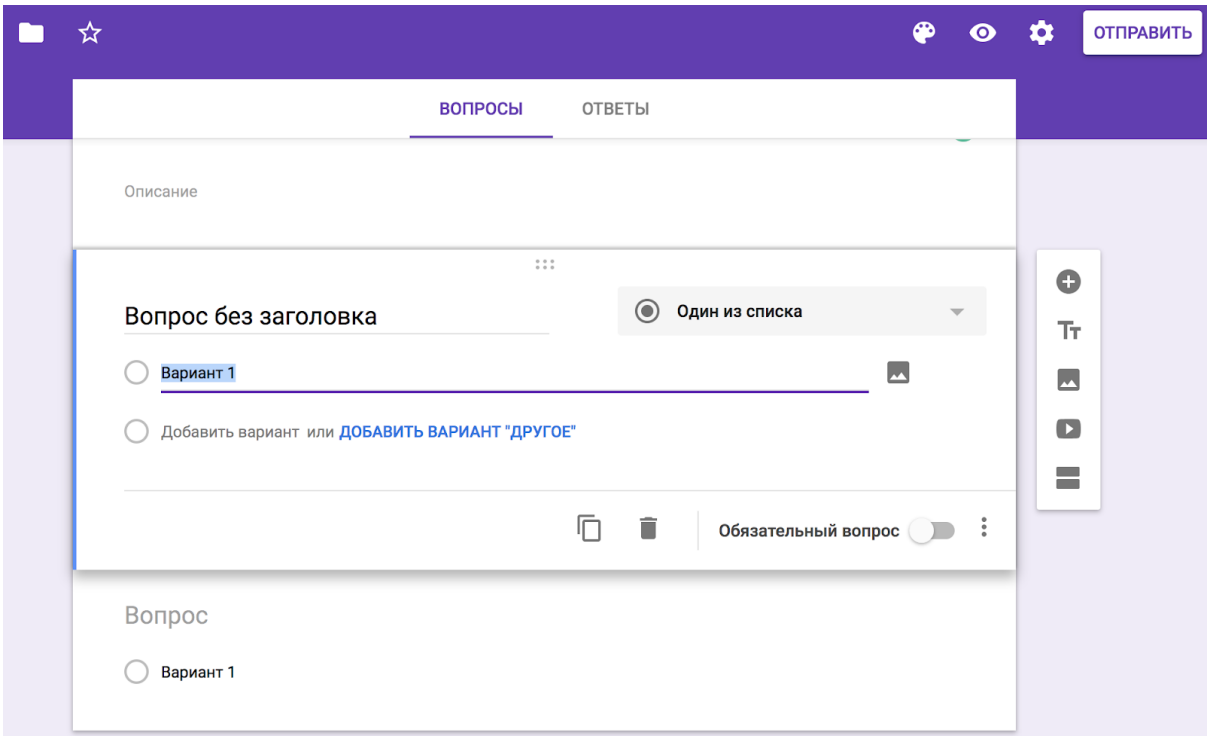
1.3 Аналіз успішних IT-проектів

Наразі на ринку існує достатня кількість програмного забезпечення для створення електронного опитування. Розберемо найпопулярніші з них.

Google forms

Google forms займає лідируюче місце серед представників даного типу додатків. Серед критеріїв успіху можна визначити відповість бренду та тісну інтеграцію з іншими додатками компанії.

Інтерфейс досить зручний, інтуїтивно зрозумілий, дизайн виконаний в стилі метеріал, з ефектом 3д ефектів в окремих компонентів. З зображенням інтерфейсу можна ознайомитися на рисунку 1.2



Рисунку 1.2 - Інтерфейст Google forms

Робоча область розділена на два блоки, переключення між якими відбувається за допомогою табів у верхній частині додатку. Працює додаток на основі SPA, що покращує відчуття користувача. В першому блоці розташований конструктор полів форми, за допомогою якого відбувається

створення та налаштування запитань. При додані нового поля користувач має можливість вибрати тип поля. Потім залежно від типу поля вибрати специфічні атрибути, наприклад опції для поля з декількома варіантами відповіді. На рисунку 1.3 зображені варіанти вибору типу поля.

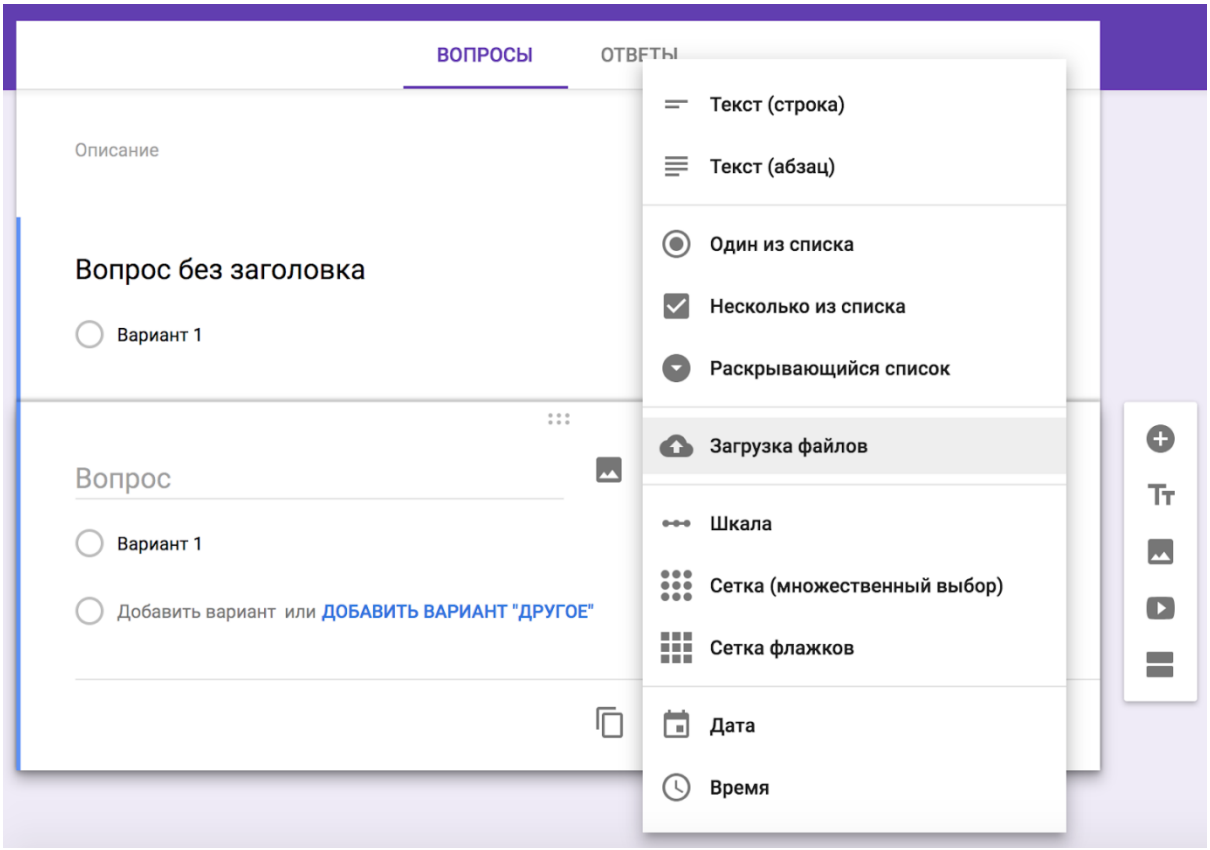


Рисунок 1.3 - Варіанти вибору типу поля

Також, є можливість вибору полів вводу для дати та часу, та групування елементів за допомогою сітки. Також цікавим функціоналом є добавлення медіа ресурсів (фото, відео) в тіло форми.

Іншим блоком є область з відповідями, де адміністратор (власник форми може переглянути) як користувачі заповнювали форму, що є досить зручним способом комбінування самої структури опитування, яку можна буде змінювати і в майбутньому та результатів цього опитування в одному місці. Окремо можна зупинитися на способах поширення опитування. Тут також досить великий вибір: через розповсюдження на електронну пошту, за

допомогою унікального URL адресу або через впровадження форми на власний сайт за допомогою відповідної розмітки.

Підсумуючи, серед основних переваг можна відмітити:

- зручний, інтуїтивно-зрозумілий інтерфейс;
- інтеграція з іншими продуктами від google;
- велика кількість інструментів розповсюдження.

Серед недоліків найбільш вагомим є відсутність функціоналу логічного поєднання полей, кожне поле є незалежним, можливість авто-заповнення через відповіді на попередні питання також відсутня.

SurveyMonkey

SurveyMonkey - наступний проект вартий уваги. Дизайн стриманий, у зеленій кольоровій гамі. Виглядає складнішим в порівняно з Google Forms, менш інтуїтивно зрозумілим.

Під час створення опитування можна вказати тематику опитування, вибрати базовий шаблон, якщо такий наявний для даної теми. Або створити з чистого листа, якщо нічого необхідного немає. При цьому вказується лише категорія опитування. Далі відбувається процес створення полів відповідної форми. Питання створюються один за одним, подібно до функціоналу Google Forms. Приклад створення поля вводу інформації можна переглянути на рисунку 1.4

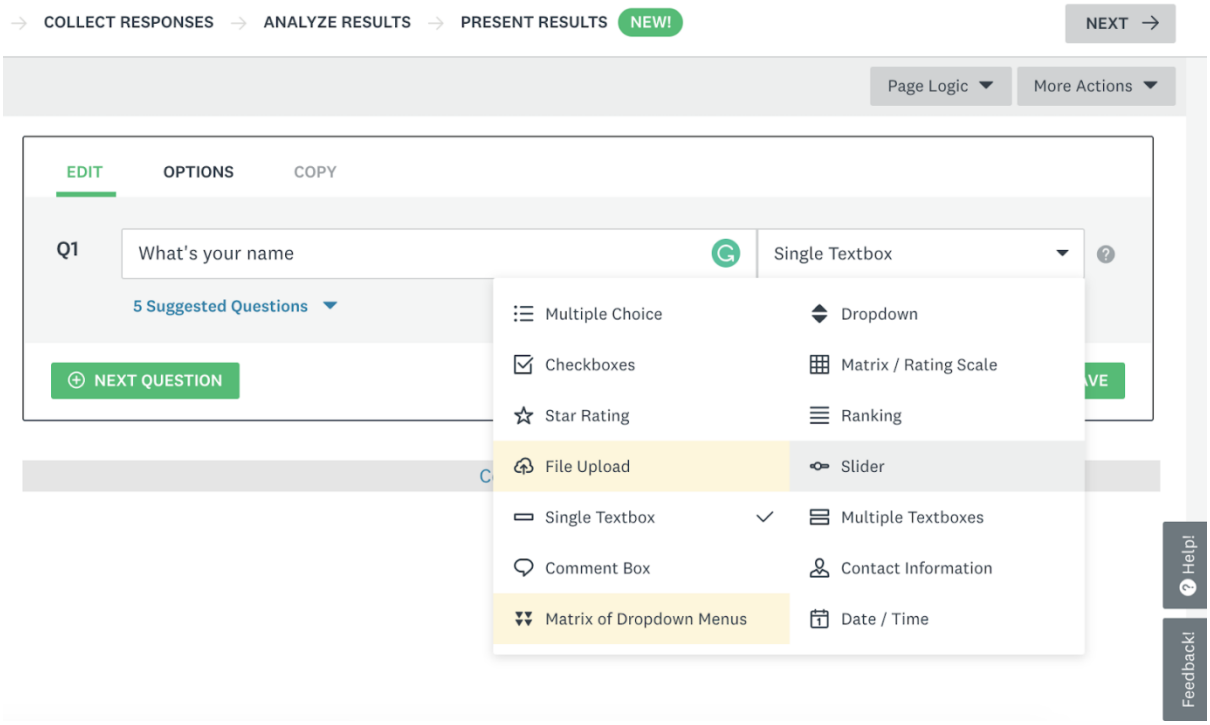


Рисунок 1.4 - Створення поля за допомогою SurveyMonkey

На відміну від Google Forms, SurveyMonkey має обмеження для безкоштовної версії. Серед основних критеріїв обмеження анкети на 10 запитань, максимальна кількість респодентів - 100. Також у платній версії є більша кількість типів полів вводу (більше 15-и) та додатковий функціонал переходів між полями (можливість базової реалізації варіативних опитувань), хоча логіка взаємодії обмежується лише переходу до вказаного питання, залежно від відповіді на деяке питання.

Отож, серед переваг:

– шаблони для створення опитувань;

– можливість переходу до конкретних питань залежно від попередніх відповідей (хоча це входить лише в платну версію).

Серед недоліків можна виділити:

– серйозні кількісні обмеження на використання безкоштовної версію (по-суті - це лише демо для платної версії);

– ускладнений функціонал.

Typeform

Typeform - також досить важливий гравець серед додатків спеціалізованих на створення опитувань. Який увібрав все найкраще з двох попередніх. Від Google Forms досить простий симпатичний інтерфейс, а від SurveyMonkey - можливість використання шаблонів, та логічне поєднання полів. До речі, в Typeform воно краще розвинуте, в плані більш гнучкої системи переходів (тут вона називається “логічними стрибками”). Але все це сховано за професійною, платною версією. В базовій існує можливість створення лише звичайних полів, які ніяк не пов’язані між собою.

Порівнявши найпопулярніші додатки, можна зробити висновок щодо важливості простоти інтерфейсу під час створення опитувань. Більшість продуктів представляють однаковий функціонал для створення логічно не пов’язаних елементів опитувань, а ті ,що мають можливість створення поєднання полів всередині опитування, приховують цю можливість за платною версією.

1.4 Аналіз вимог до програмного забезпечення

Перед визначенням вимог до програмного забезпечення слід визначити дійових осіб для системи опитування.

Серед головних учасників слід виділити наступних дійових осіб:

- адміністратор системи опитування (власник опитування);
- користувач системи опитування (респодент).

При чому користувачі можуть бути двох типів:

- авторизовані користувачі;
- аноніми (неавторизовані користувачі).

Серед основних вимог залежно від дійових осіб для адміністратора можна виділити наступні можливості:

- авторизуватися в системі;
- створювати опитування;

- оновлювати опитування;
- видалити опитування;
- переглядати результати опитувань;
- надіслати створене опитування користувачу.

В свою чергу користувачам мають бути доступні такі можливості:

- зареєструватися;
- авторизуватися, для зареєстрованих користувачів;
- переглянути опитування;
- заповнити опитування;
- надіслати результати опитування.

При цьому користувачі можуть також бути адміністраторами, створивши власне опитування.

1.4.1 Розроблення функціональних вимог

Серед функціональних вимог, яким повинна відповідати система можна виділити наступні варіанти використання, попередньо розділивши їх залежно від дійової особи та представивши їх у вигляді сценаріїв використання.

Сценарії використання для адміністратора представлені в таблиці 1.1

Таблиця 1.1 - Сценарій використання UC-001

Назва	Авторизація адміністратора
Опис	Адміністратор може авторизуватися
Дійові особи	Адміністратор (користувач, що створив власне опитування)
Передумови	Адміністратор вийшов з системи
Постумови	Адміністратор увійшов в систему
Основний сценарій	Адміністратор заповнює приватну інформацію для входу в систему

Продовження таблиці 1.1

Розширений сценарій	Адміністратор при вході в систему автоматично перенаправляється на сторінку авторизації, де необхідно ввести дані логіну та паролю. Після чого натискає на кнопку “далі” та переходить у свій кабінет при коректності даних, або отримує повідомлення про помилку у іншому випадку.

Таблиця 1.2 - Сценарій використання UC-002

Назва	Перегляд списку опитувань
Опис	Адміністратор може переглядати список опитувань
Дійові особи	Адміністратор
Передумови	Адміністратор пройшов авторизацію
Постумови	Відображення списку опитувань
Основний сценарій	Авторизований адміністратор може переглядати створені опитування
Розширений сценарій	Адміністратор успішно пройшовши авторизацію перенаправляється в особистий кабінет, де відображається список створених опитувань, або повідомлення що таких не існує

Таблиця 1.3 - Сценарій використання UC-003

Назва	Створення опитування
Опис	Адміністратор може створювати опитування

Продовження таблиці 1.3

Дійові особи	Адміністратор
Передумови	Адміністратор пройшов авторизацію
Постумови	Створено нове опитування
Основний сценарій	Адміністратор, заповнивши форму в конструкторі, може створити нове опитування
Розширений сценарій	Адміністратор з особистого кабінету може перейти до конструктора опитування, де створюючи поля одне за одним може сформувати форму опитування. Також необхідно заповнити поля з назвою та описом опитування, після чого натиснути кнопку ‘створити’.

Таблиця 1.4 - Сценарій використання UC-004

Назва	Попередній огляд створюваного опитування
Опис	Адміністратор має можливість переглядати форму під час налаштування її полів
Дійові особи	Адміністратор
Передумови	Адміністратор почав створювати нове опитування
Постумови	Відображення створюваної форми
Основний сценарій	При конфігурації поля відбувається відображення форми

Продовження таблиці 1.4

Розширений сценарій	Після добавлення нового поля у форму опитування, поле миттєво відображається у результуючій формі (яка знаходиться справа від конструктора форми), при зміні параметрів поля (додання нових атрибутів, опцій) всі зміни також відображаються у результуючій формі

Таблиця 1.5 - Сценарій використання UC-005

Назва	Видалення полів під час створення форми
Опис	Адміністратор може видаляти добавлені поля
Дійові особи	Адміністратор
Передумови	Адміністратор почав створення опитування та добавив у форму поля
Постумови	Видалення поля з створюваної форми
Основний сценарій	Адміністратор може видалити поле натиснувши на кнопку видалення
Розширений сценарій	Адміністратор, вибравши необхідне для видалення поле, робить його активним, та за натиснувши на кнопку “видалити” видаляє його з опитування

Таблиця 1.6 - Сценарій використання UC-006

Назва	Модифікація полів під час створення
Опис	Адміністратор може модифікувати створені поля

Продовження таблиці 1.6

Дійові особи	Адміністратор
Передумови	Адміністратор почав створення опитування та добавив у форму поля
Постумови	Модифікація поля з створюваної форми
Основний сценарій	Адміністратор може модифікувати атрибути поля, вказавши його назву, тип, значення за замовчуванням
Розширений сценарій	Адміністратор може модифікувати атрибути добавленого поля. Для цього необхідно обрати відповідне поле із загального списку полів, щоб зробити його активним. Щойно додані поля автоматично стають активними. Потім через введення даних в поля “назва”, “тип поля”, “значення” адміністратор змінює значення цих атрибутів

Таблиця 1.7 - Сценарій використання UC-007

Назва	Модифікація стану полів під час створення
Опис	Адміністратор може модифікувати стан створених полів
Дійові особи	Адміністратор
Передумови	Адміністратор почав створення опитування та добавив у форму поля
Постумови	Модифікація стану поля з створюваної форми
Основний сценарій	Адміністратор може модифікувати атрибути стану поля, вказавши його видимість, активність, значення

Продовження таблиці 1.7

Розширений сценарій	Адміністратор може модифікувати атрибути стану доданого поля. Для цього необхідно активувати відповідне поле. Потім через введення даних в поля “видимість”, “активність”, “значення”, які знаходяться всередині блоку “стан” адміністратор змінює значення цих атрибутів стану поля

Таблиця 1.8 - Сценарій використання UC-008

Назва	Автоматична модифікація стану полів
Опис	Адміністратор може створювати автоматичну модифікацію стану полів через зміни інших полів
Дійові особи	Адміністратор
Передумови	Адміністратор почав створення опитування та добавив у форму поля
Постумови	Автоматичну модифікація станів полів через зміни інших полів
Основний сценарій	Адміністратор може створювати автоматичну модифікацію стану полів прив’язавши значення інших полів до атрибутів стану

Продовження таблиці 1.8

Розширений сценарій	Адміністратор може прив’язувати значення атрибутів стану поля до значення інших полів. Зміна значення зв’язаних полів призведе до автоматичної зміни значення атрибутів стану прив’язаного поля. Прив’язування відбувається через використання назви поля всередині значення атрибуту стану. Замість назви буде автоматично використовуватися значення цього поля. Наприклад, ввівши як значення атрибуту стану поля “поле2” рядок “поле1 + 1”, значення другого поля буде автоматично збільшуватися на 1 при модифікації першого поля

Таблиця 1.9 - Сценарій використання UC-009

Назва	Використання математичних операцій для автоматична модифікації стану полів
Опис	Адміністратор може використовувати математичні операції для автоматичної модифікації стану полів
Дійові особи	Адміністратор
Передумови	Адміністратор почав створення опитування та добавив у форму поля
Постумови	Автоматичну модифікація станів полів з використанням математичних операцій
Основний сценарій	Адміністратор може створювати автоматичну модифікацію стану полів з використанням математичних операцій

Продовження таблиці 1.9

Розширений сценарій	Після прив'язування значення стану поля до значення іншого поля адміністратор може використовувати математичні операції для зміни стану. Наприклад, ввівши як значення атрибуту стану поля "полеБ" рядок "полеА * 4", значення другого поля буде автоматично збільшуватися в 4 рази залежно від першого поля. Серед операндів може бути декілька полів. Але не допускається використання в тілі атрибуту стану поля звертання до цього ж поля, для уникнення рекурсивних залежностей. Серед математичних операції доступні наступні: – додавання/віднімання; – множення/ділення; – піднесення до степеня; – обчислення квадратного корню від числа.

Таблиця 1.10 - Сценарій використання UC-010

Назва	Використання логічних операцій для автоматична модифікації стану полів
Опис	Адміністратор може використовувати логічні операції для автоматичної модифікації стану полів
Дійові особи	Адміністратор
Передумови	Адміністратор почав створення опитування та добавив у форму поля
Постумови	Автоматичну модифікація станів полів з використанням логічних операцій

Продовження таблиці 1.10

Основний сценарій	Адміністратор може створювати автоматичну модифікацію стану полів з використанням логічних операцій
Розширений сценарій	Після прив'язування значення стану поля до значення іншого поля адміністратор може використовувати логічні операції для зміни стану. Наприклад, ввівши як значення атрибуту відображення стану поля "полеБ" рядок "полеА" друге поле буде відображатися лише після заповнення першого. Серед операндів може бути декілька полів. Серед логічних операції доступні наступні: – або; – і; – більше/менше; – дорівнює/не дорівнює; – входить в перелік.

Перейдемо до функціональних вимог зі сторони користувача.

Таблиця 1.11 - Сценарій використання UC-011

Назва	Реєстрація користувача
Опис	Користувач може зареєструватися
Дійові особи	Користувач
Передумови	Користувач зайшов в додаток
Постумови	Користувач зареєструвався в систему
Основний сценарій	Користувач заповнює приватну інформацію для реєстрації в системі

Продовження таблиці 1.11

Розширений сценарій	Користувач при вході в систему автоматично перенаправляється на сторінку авторизації, звідки він може перейти на сторінку реєстрації, де необхідно ввести дані ім’я, прізвища, пошти, паролю. Після чого підтвердити пароль ще раз та натиснути на кнопку “далі”. Після цього користувач може авторизуватися.

Таблиця 1.12 - Сценарій використання UC-012

Назва	Авторизація користувача
Опис	Користувач може авторизуватися
Дійові особи	Користувач
Передумови	Користувач вийшов з системи
Постумови	Користувач увійшов в систему
Основний сценарій	Користувач заповнює приватну інформацію для входу в систему
Розширений сценарій	Користувач при вході в систему автоматично перенаправляється на сторінку авторизації, де необхідно ввести дані полів логіну та паролю. Після чого натискає на кнопку далі та переходить у свій кабінет при коректності даних, або отримує повідомлення про помилку у іншому випадку.

Таблиця 1.13 - Сценарій використання UC-013

Назва	Проходження опитування користувачем

Продовження таблиці 1.13

Опис	Користувач може проходити опитування
Дійові особи	Користувач
Передумови	Користувачу було надіслане опитування
Постумови	Користувач пройшов опитування
Основний сценарій	Користувач дав відповіді на запитання опитування
Розширений сценарій	Перейшовши по надісланому посиланню, користувач отримав форму на заповнення. Заповнивши всі поля форми та натиснувши кнопку “далі” користувач надіслав свої відповіді адміністратору форми

Даний проект має наступні функціональні вимоги:

Таблиця 1.15 - Опис функціональної вимоги REQ-001

Номер	REQ-001
Назва	Реєстрація користувача
Опис	Користувач може зареєструватися в системі

Таблиця 1.16 - Опис функціональної вимоги REQ-002

Номер	REQ-002
Назва	Авторизація користувача
Опис	Користувач може авторизуватися в системі

Таблиця 1.17 - Опис функціональної вимоги REQ-003

Номер	REQ-003
Назва	Перегляд створених опитувань
Опис	Користувач може переглядати створені опитування

Таблиця 1.18 - Опис функціональної вимоги REQ-004

Номер	REQ-004
Назва	Створення опитувань
Опис	Користувач може створювати опитування

Таблиця 1.19 - Опис функціональної вимоги REQ-005

Номер	REQ-005
Назва	Оновлення опитувань
Опис	Користувач може оновлювати опитування

Таблиця 1.20 - Опис функціональної вимоги REQ-006

Номер	REQ-006
Назва	Заповнення опитувань
Опис	Користувач може заповнювати опитування

Таблиця 1.21 - Опис функціональної вимоги REQ-007

Номер	REQ-007
Назва	Перегляд результатів
Опис	Користувач може перегядати результати опитування

Матриця трасування зображена на рисунку 1.5

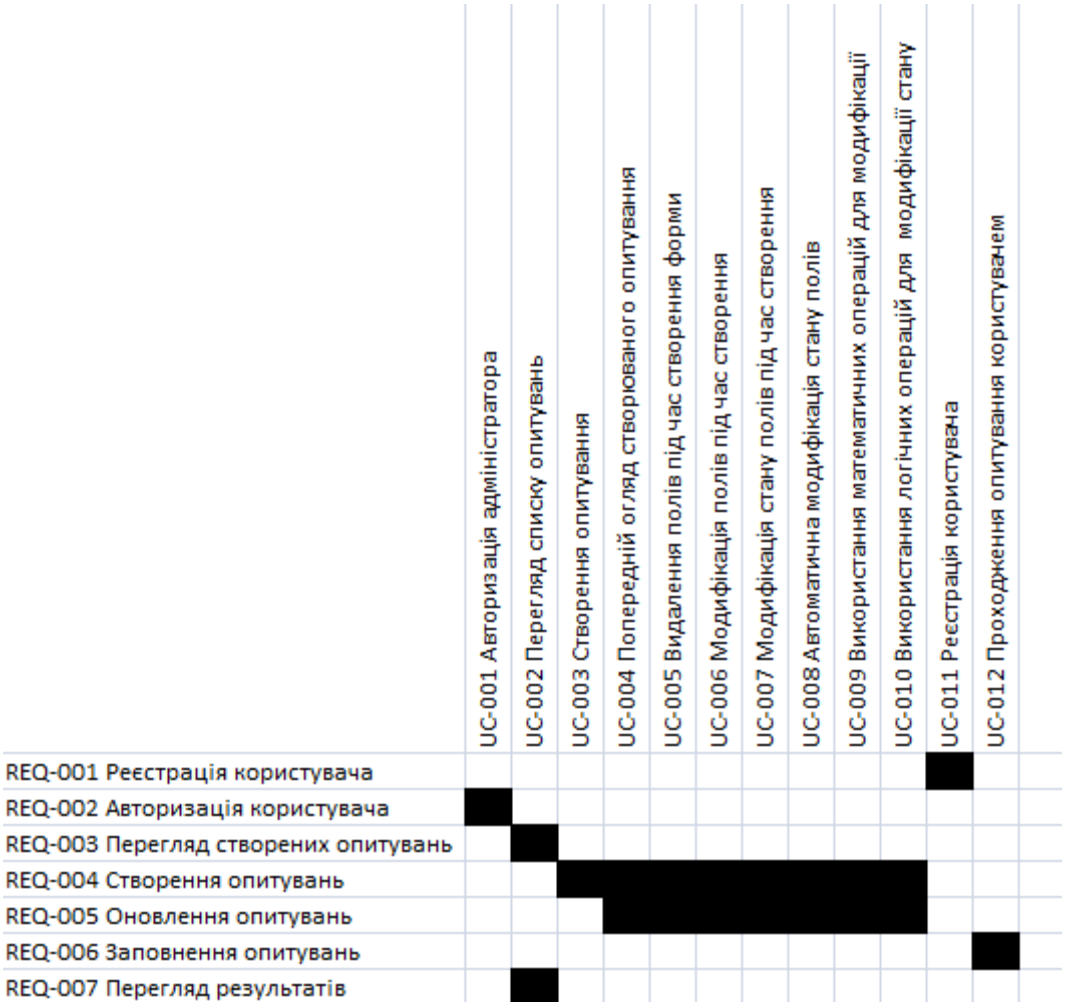


Рисунок 1.5 – Матриця трасування

1.4.2 Розроблення нефункціональних вимог

Система автоматизації варіативних опитувань повинна відповідати наступним нефункціональним вимогам:

- виконано у вигляді SPA додатку;
- підтримка усіма сучасними браузерами (2 останні версії), доля ринку яких більша за 0.7%;
- підтримка екранів з усіма розширеннями;
- передача даних за допомогою REST архітектури;
- розмір бандлу не перевищує 1 мб javascript коду.

1.4.3 Постановка комплексу завдань модулю

Розробка системи автоматизації варіативних опитувань призначене для полегшення створення складних опитувань з логічно зв’язаними блоками питань.

Метою роботи - є полегшення процесу створення опитувань з логічно зв’язаними структурними блоками. Підвищення зручності їх використання як зі сторони адміністратора, та і зі сторони користувача за допомогою розробки одно-сторінкового веб-додатку, із збереженням даних на серверній частині.

Для цього система повинна надавати можливість виконувати наступні завдання:

- реєстрація користувачів;
- авторизація в системі;
- перегляд створених опитувань;
- створення опитування;
- оновлення опитування;
- видалення опитування ;
- перегляд результату опитування;
- розповсюдження створених опитувань серед користувачів;
- перегляд опитування;

- заповнення опитування;
- надсилання заповненого опитування.

1.5 Висновки по розділу

Опитування - важливий соціальний інструмент збору інформації, який широко увійшов у наше повсякденне життя. Окремим випадком є розгалуженні опитування, які характеризуються збільшеною складністю структури питань. Питання варіативних опитувань мають логічні залежності від значень інших полів. Саме ця група опитувань найменш реалізована серед існуючих IT-рішень, а ті, у яких реалізована можливість створення варіативних опитувань, приховують її за платною версією продукту та містять інші недоліки. Що є підставою для розробки власного проекту із метою вирішення даних проблем.

Найбільш привабливою формою продукту із сторони зручності користування є вед додатки, а точніше SPA. Адже вони не потребують встановлення додаткового програмного забезпечення, а працюють прямо з браузера. Таким чином проблеми кросплатформеності лягають на сторону розробників самих браузерів. Іншою причиною зручності SPA є принцип їх роботи - один раз завантажившись, вони працюють під час всього етапу комунікації з додатком, імітуючи повноцінний десктопний додаток.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Для моделювання програмного забезпечення будемо використовувати BPMN діаграми. [4]

Почнемо з клієнтської частини.

2.1.1 Клієнтська частина

Для реєстрації користувача необхідно заповнити поля з приватною інформацією, які включають прізвище та ім’я користувача, логін, електронну пошту та пароль. Далі натиснувши кнопку “далі” відбувається запит на сервер. Після обробки надісланої інформації сервер відповідає або повідомленням про успішну реєстрацію, або повідомлення про помилку. Детальніше процес реєстрації описаний на рисунку 2.1

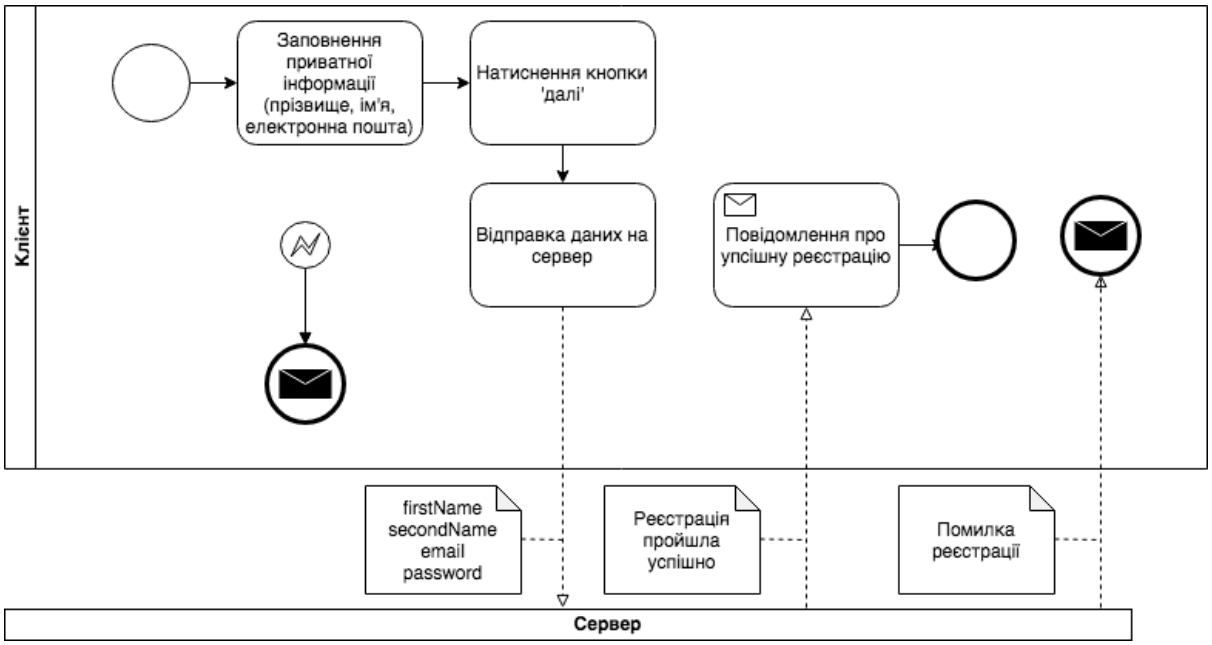


Рисунок 2.1 - Реєстрація користувача на клієнтській частині

Наступний крок взаємодії з додатком є авторизація. Для авторизації користувачу необхідно ввести свою електронну пошту та пароль, після чого користувач, натиском на кнопку ‘далі’, робить запит на серверну частину для отримання токену авторизації якщо дані були вірні, або повідомлення про помилку в іншому випадку. Схема процесу зображена на рисунку 2.2

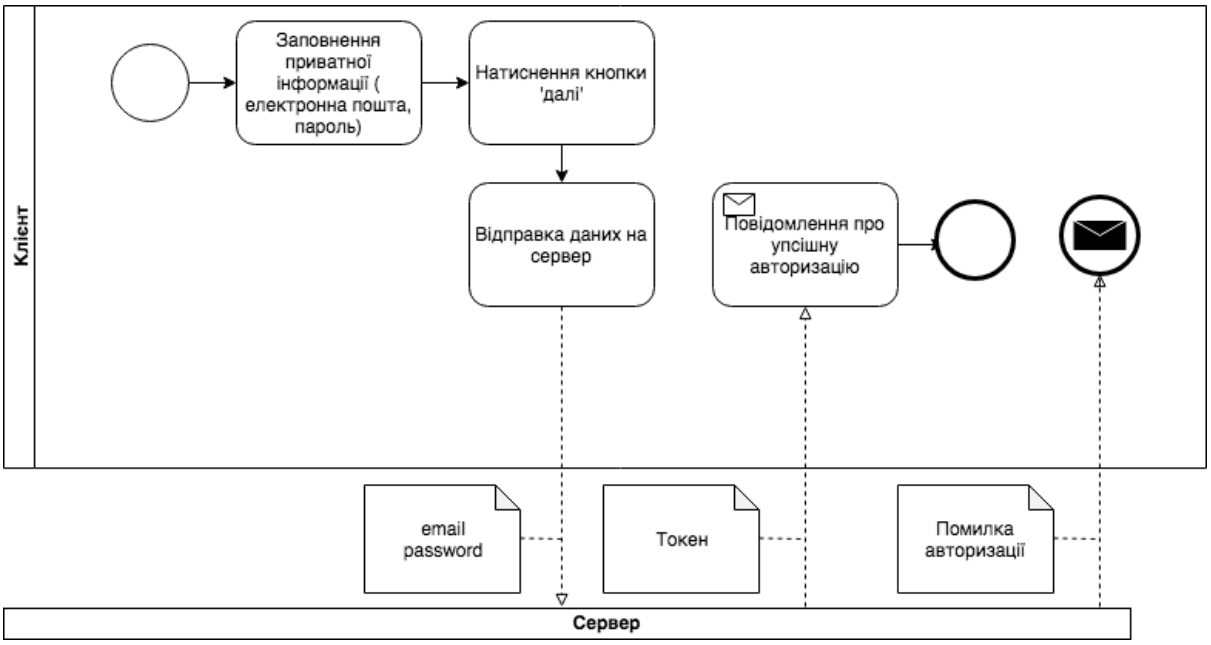


Рисунок 2.2 - Процес авторизації на клієнтській частині

Після успішної авторизації користувач перенаправляється у особистий кабінет, де розміщується список створених ним опитувань, якщо такі наявні. З особистого кабінету користувач може перейти в конструктор опитувань, натиснувши кнопку ‘створити’. Після перенаправлення в конструктор користувачу відображається панель керування, за допомогою, якої можна створювати нові питання, видаляти добавлені питання, а також змінювати конфігурацію питань. Після створення питань опитування необхідно вказати назву опитування та короткий опис. Після натиснення кнопки ‘далі’ відбудеться запит на серверну частину із передачею всіх заповнених полів. Детальний опис створення нового опитування відображений на рисунку 2.3

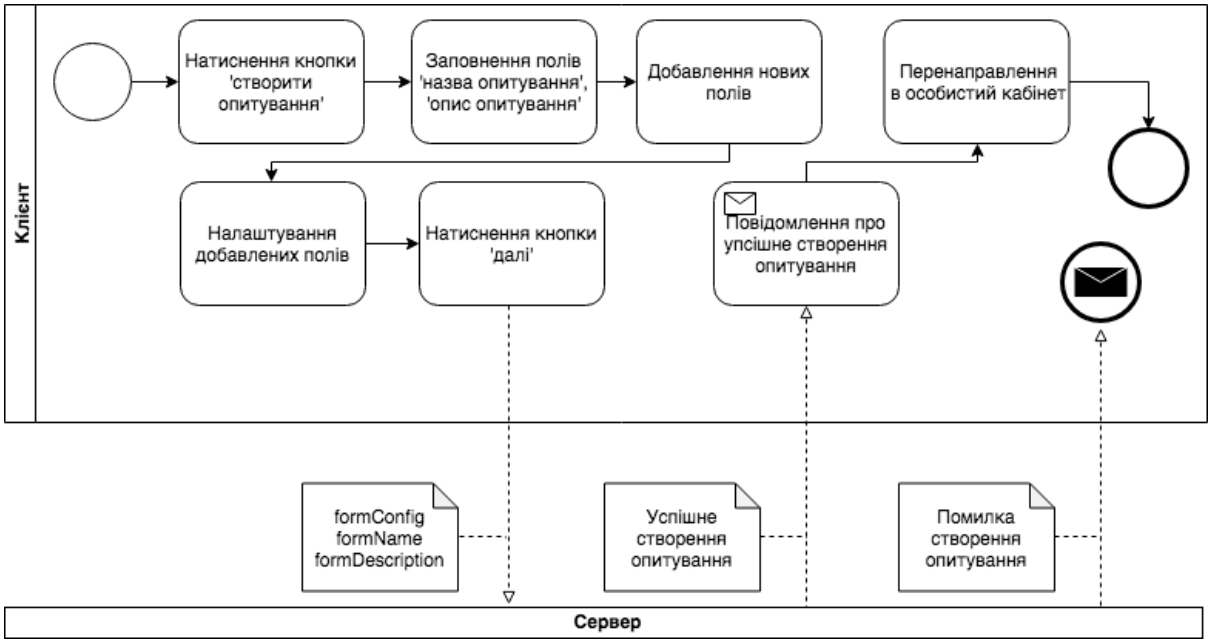


Рисунок 2.3 - Процес створення нового опитування на клієнтській частині

Для модифікації добавленого питання в опитуванні необхідно активувати необхідне поле, вибором його серед загального списку полів (щойно додані поля автоматично стають активними), після чого внести відповідні зміни в атрибути поля через панель управління конструктора форми. Після внесених змін натиснути кнопку ‘далі’. Детальний опис процесу зображений на рисунку 2.4

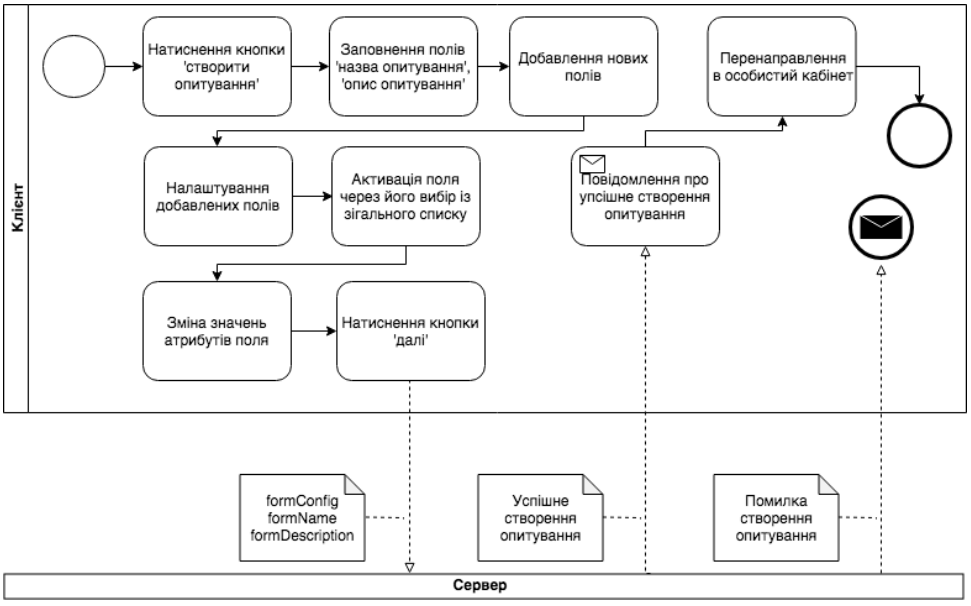


Рисунок 2.4 - Оновлення атрибутів добавленого поля

2.1.2 Серверна частина

Реєстрація користувача відбувається після отримання запиту від клієнта з передачею його особистих даних, які включають прізвище та ім’я користувача, електронну пошту користувача та пароль, після чого сервер вносить ці дані до бази даних, з перевіркою на існування користувача з такою ж поштою. Якщо таких користувачів не знайдено, реєстрація проходить успішно. В іншому випадку на клієнтську частину відправляється повідомлення про помилку. Опис процесу можна знайти на рисунку 2.5

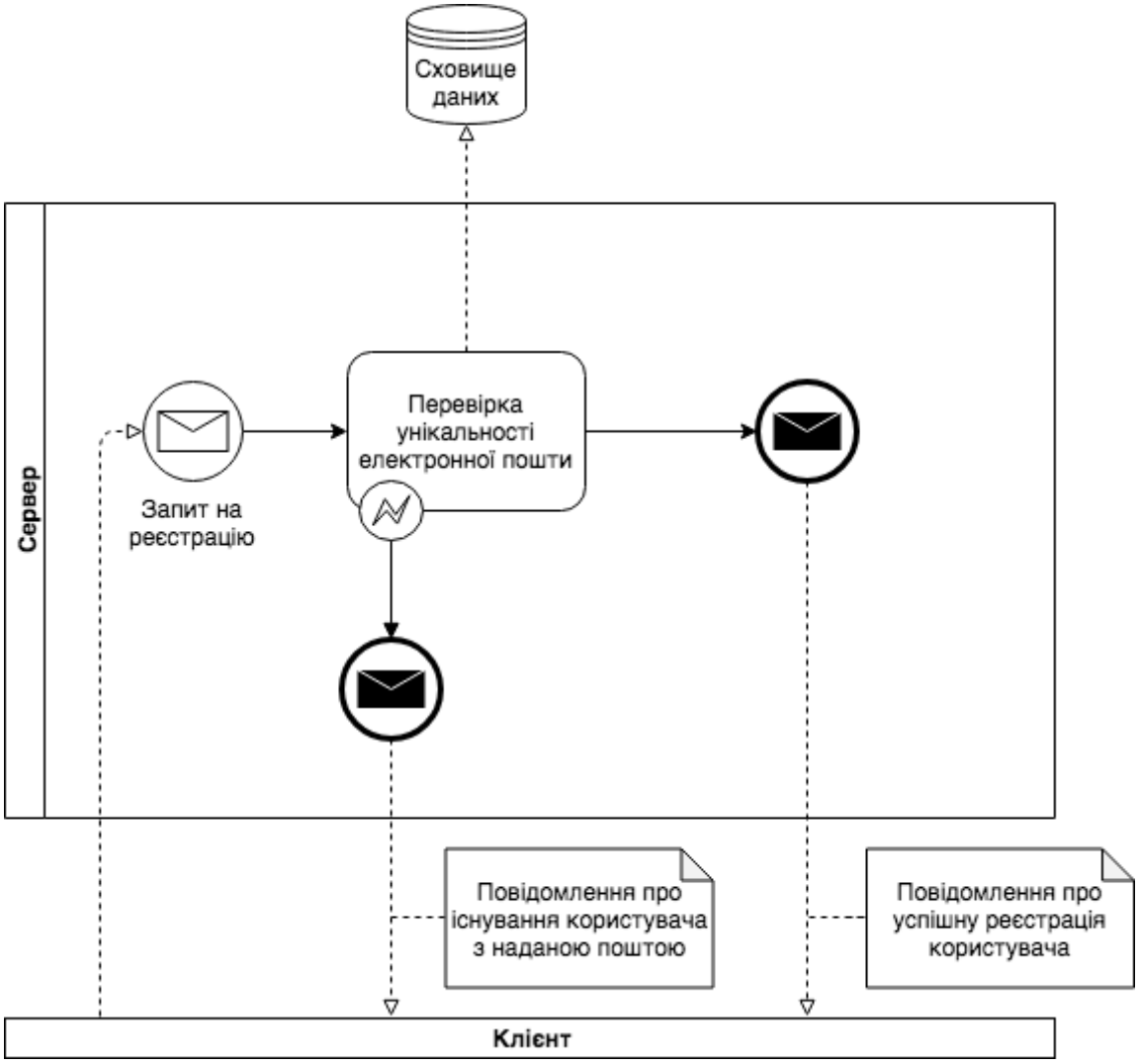


Рисунок 2.5 - Реєстрація користувача на серверній частині

Авторизація користувача відбувається після отримання запиту з клієнтської частини. Разом з запитом приходять дані користувача: електронна пошта та пароль. Після цього необхідно перевірити існування такого користувача, та у випадку успішної перевірки згенерувати токен, в іншому випадку надіслати повідомлення про помилку. Опис процесу зображений на рисунку 2.6

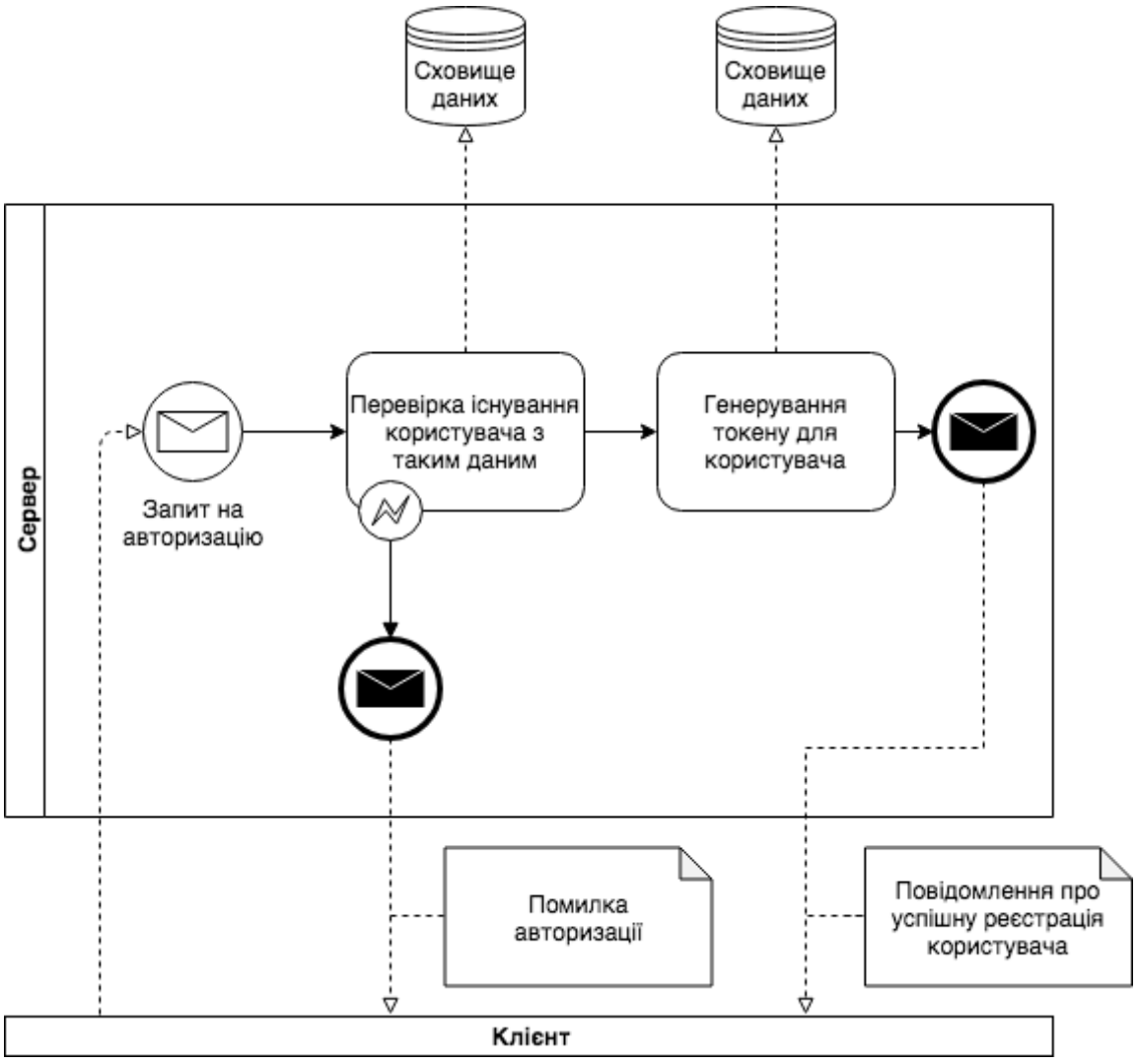


Рисунок 2.6 - Авторизація користувача на серверній частині

Після отримання запиту від клієнтської частини на передачу списку опитувань, створених даним користувачем, необхідно перевірити стан авторизації користувача. Якщо користувач авторизований необхідно зробити вибірку опитувань створених цим користувачем. Та у разі відсутності помилок надіслати список на клієнтську частину. У іншому ж випадку надіслати

повідомлення про помилку. Процес вибірки опитувань відображений на рисунку 2.7

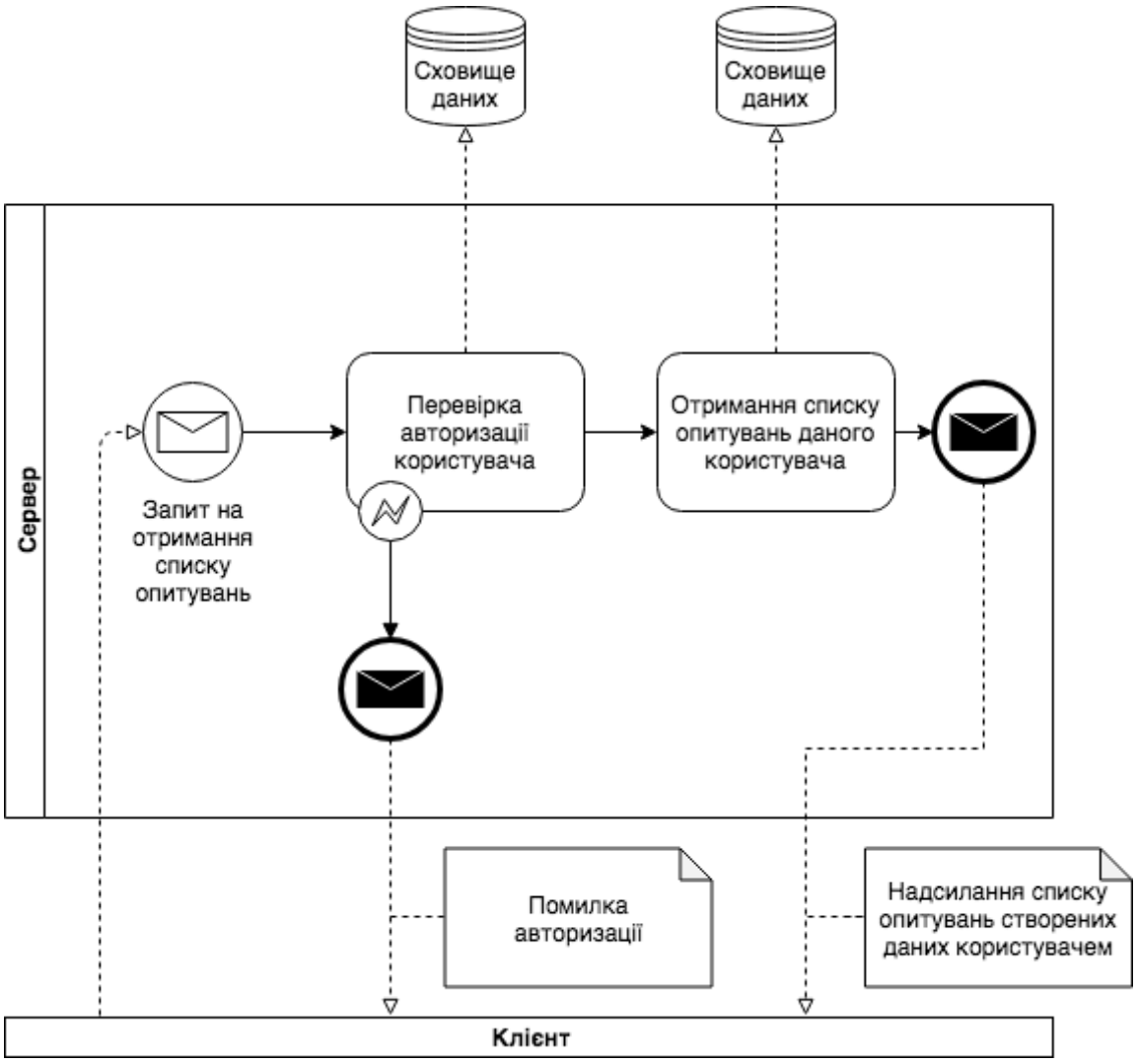


Рисунок 2.7 - Процес вибірки опитувань на серверній частині

Для створення або модифікування опитування, необхідно також перевірити авторизацію користувача, після цього провести валідацію отриманих даних, та зберегти дані щодо опитування у сховище даних, закріпивши дане опитування за цим користувачем. У разі помилки авторизації, валідації або інших внутрішніх помилок надіслати відповідне повідомлення на клієнтську частину. З описом процесу можна ознайомитися на рисунку 2.8

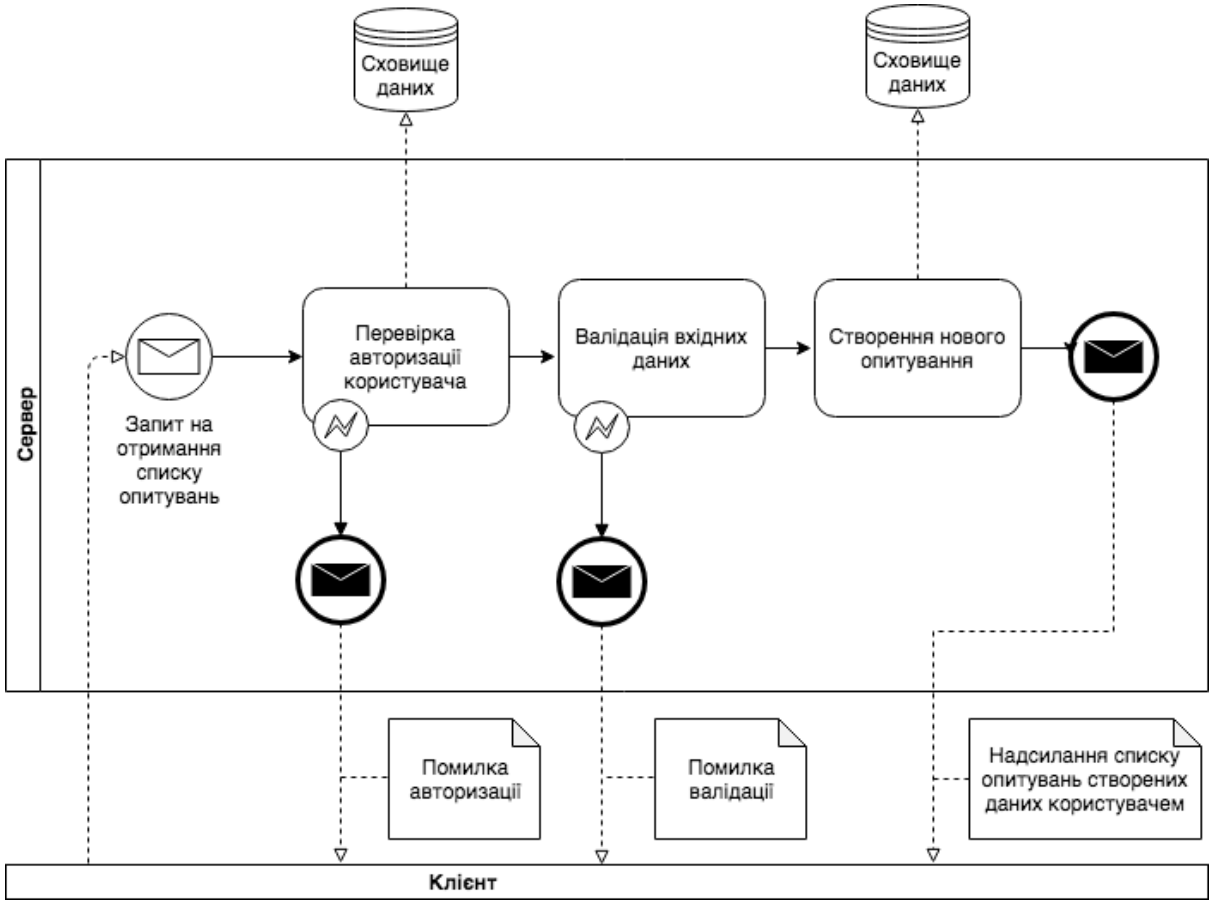


Рисунок 2.8 - Процес створення нового опитування на серверній частині

2.2 Архітектура програмного забезпечення

Для реалізації вирішення комплексу задач з автоматизації варіативних опитувань було прийнято рішення розроблення бібліотеки для побудови динамічних форм на клієнтській частині. Необхідність розробки окремої бібліотеки полягала у декількох аспектах. По-перше, форми є ключовим елементом система. Саме через взаємодію з формами відбувається проходження опитування. Також під час створення адміністратор повинен мати можливість переглянути результат. Тому, правильним, з точки зору архітектури, рішенням буде відокремлення логіки побудови форми у окрему бібліотеку, яка зможе функціонування і як окреме рішення для вирішення схожих проблем, з додаванням власного функціоналу. Даний підхід дає змогу розширити можливості розробки просто програмного продукту для вирішення

конкретної проблеми, а дійсно представляє комплекс задач для автоматизації процесів у цій категорії проблем.

Щодо самої бібліотеки є ряд вимог. Вона має забезпечувати ефективне відображення форми, з багатьма полями, які мають можливість взаємодії між собою. Оскільки метою є побудова веб-додатку, середовищем роботи клієнтської частини буде браузер, а дана задача вимагає досить частого процесу перебудови документо-об’єктної моделі, існує необхідність використання ефективних рішень для усунення проблем з швидкодією. Одним з найефективніших рішень є використання технологій, основаних на віртуалізації документо-об’єктної моделі. Суть її роботи полягає в оперуванням об’єктами не на рівні документно -об’єктної моделі, а на рівні javascript об’єктів, які при необхідності синхронізуються з документо-об’єктною моделлю для відображення необхідного стану об’єктів. Флагманським рішенням наразі є бібліотека React, яка розроблена департаментом досліджень та розробки компанії Facebook, яке ми і використаємо для розробки нашої власної бібліотеки. Для легкого перевикористання форм, побудованих за її допомогою, необхідно спроектувати зручний спосіб задання вхідних даних. Оскільки, основною мовою розробки веб-клієнтів у браузері є javascript, а її найгнучкішим типом даних є об’єкти, доцільно реалізувати вхідні параметри форми за допомогою одного об’єкту, який буде містити основну інформацію про дану форму та набір полів у вигляді масиву конфігураційних об’єктів. Так це нам дасть змогу простої та очевидної передачі інформації про форму у вигляді серіалізованих об’єктів та простого збереження даних JSON формату у сховищі.

Що стосується логіки роботи форми, основною її особливість є саме можливість одних полів впливати на поведінку інших. При цьому потрібно вирішити питання передачі повідомлень між залежними компонентами, питання використання рекурсивних залежностей та ін. Вирішення цього типу задач полягає у використанні шаблону проектування “Наглядач”. [5]

Його принцип полягає у тому, що батьківський компонент (компонент, від якого залежать інші компоненти) має інформацію про нащадків (компоненти, які залежать від батьківського), і при оновленні значення чи стану батьківського, він автоматично розповсюдить усім нащадкам відповідні повідомлення. Сформувавши деревовидну структуру, можна прийти до рішення, де батьківський компонент відносно своїх нащадків може бути нащадком інших компонентів, при цьому повідомлення найвищих в ієрархії компонентів будуть проходити на декілька рівнів униз. Саме, дана структура дасть змогу побудови складних варіативних форм.

Спрощена схема взаємодії компонентів зображена на рисунку 2.9

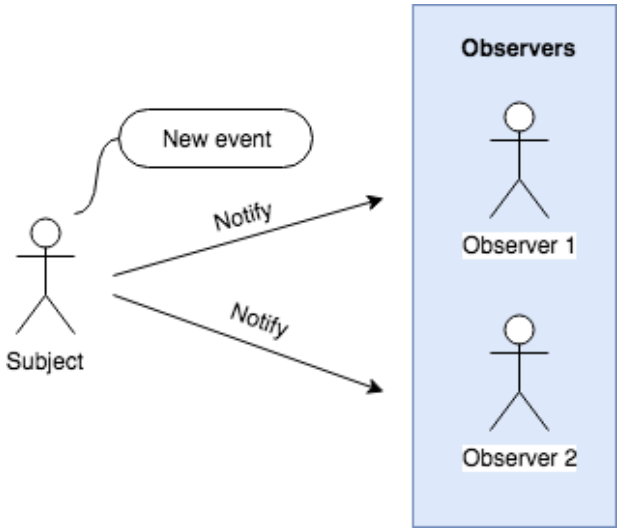


Рисунок 2.9 - Шаблон проектування “Спостерігач”

Щодо самого прикладного програмного проекту, він буде побудований використовуючи клієнт-серверну архітектуру. Клієнт буде використовувати бібліотеку описану вище, для побудови форм. Для побудови елементів інтерфейсу та їх взаємодії також буде використовуватися бібліотека React. [6] Клієнтська частина буде розділена на модулі, кожен з яких матиме свій унікальний адрес, для цього необхідно реалізувати роутер. Для збереження інформації на клієнті в процесі його роботи та передачі її між модулями слід використати менеджер стану, який буде імплементувати архітектуру Flux.

Схема архітектури Flux зображена на рисунку 2.10. Її суть полягає в тому, що всі дані про стан додатку зберігаються в одному місці, і змінюються внаслідок дій зі сторони презентаційної частини, які проходять етап взаємодії з відправником змін в загальний стан.

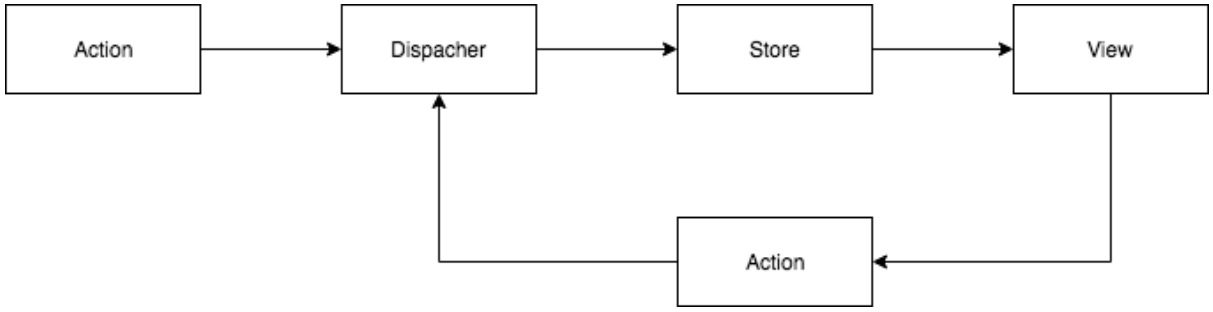


Рисунок 2.10 – Схема архітектури Flux

Як приклад імплементації цієї архітектури можна використати бібліотеку Redux.

Щодо серверної частини, вона буде побудована на платформі Node.js, з використанням серверу на Express. Для збереження даних використовуватиметься база даних MongoDB. Взаємодія клієнту та серверу буде відбуватися за допомогою REST API, де для кожного типу запиту буде використовуватися спеціальний метод HTTP запиту з передачею відповідних параметрів.

2.3 Конструювання програмного забезпечення

2.3.1 Конструювання бібліотеки побудови динамічних форм

Головний клас бібліотеки - LiveForm

Методи класу LiveForm представлені у таблиці 2.1

Таблиця 2.1 – Методи класу LiveForm

Назва метода	Аргументи	Значення, яке повертається	Призначення методу
getInitialState	Props	state	Формування початкового значення стану форми
setLiveFormFields	Props	-	Встановлення полів форми
getCurrentFormState	-	formState	Повернення стану форми
firstFieldsUpdate	formState	-	Перше оновлення полів форми
callSubscribers	subscribers, formState	-	Оновлення залежних полів
updateFormState	newFieldPart, callback	newFormState	Оновлення стану форми
onChangeFormField	fieldConfig, propName, propValue	-	Оновлення поля форми
formSubmit	-	formState	Повертає значення форми, при кінцевій відправці даних

Більшість з цих методів використовують допоміжні функції, які зібрані в окремий модуль ‘Helpers’. Методи модуля Helpers представлені у таблиці 2.2

Таблиця 2.2 – Методи модуля Helpers

Назва метода	Аргументи	Значення, яке повертається	Призначення методу
formConfigValidation	formConfig	isValidField	Валідація конфігу форми
getFormComponents	formState, liveFormFields, onChangeFormField, onSubmit	fields	Генерація полів форми на базі конфігураційного об’єкту
getFormItemByFieldType	fieldType	formItem	Повернення компоненту залежно від вхідного типу
getInitialFormState	formConfig	initialFormState	Генерація початкового стану форми
getLiveFormFields	formFields	dataSource	Добавлення до полів форми автоматичного оновлення

Продовження таблиці 2.2

addFieldsSubscribers	liveFormFields, dataSource	-	Добавлення по полів відомості про залежні компоненти
addFieldsUpdateFunc	liveFormFields, dataSource	-	Добавлення по полів функцій їх оновлення
findFieldParents	liveFormFields, stateField	parents	Знаходження батьківських компонентів поля
updateFieldByValueExpr	liveFormFields, fieldName, stateFieldName, stateField	-	Генерація функції автооновлення

2.3.2 Конструювання клієнтської частини

Клієнтська частина поділена на модулі:

- App - головний модуль додатку;
- Auth - модуль реєстрація/авторизації користувачів;
- Dashboard - особистий кабінет користувача;
- FormCreator - модуль створення нових опитувань;
- FormFiller - модуль заповлення опитувань;
- FormViewer - модуль перегляду створених опитувань та відповідей на Них;
- LiveFormBuilder - модуль генерації форм.

Кожен з модулів на верхньому рівні представляє клас, який наслідує клас Component з бібліотеки React.

Для їх взаємодії використовується бібліотека Redux, яка є імплементацією архітектури Flux. Тому, окрім даних модулів в додатку також присутні модулі для керування станом додатку.

А саме:

- Actions - дії, які будуть виконувати ініціалізацію оновлення стану;
- Redusers - функціх, для оновлення стану;
- Sagas - функції для вирішення асинхронних дій;
- Store - конфігуратор стану.

Детально дії описані у таблиці 2.3.

Таблиця 2.3 - Опис дій

Назва дії	Тип	Допоміжні дані	Призначення
getYourForms	GET_YOUR_FORMS	id	Запит на отримання форми
gotYourForms	GOT_YOUR_FORMS	data	Завершення отримання форми
getCurrentForm	GET_CURRENT_FORM	id	Запит на отримання поточної форми
gotCurrentForm	GOT_CURRENT_FORM	data	Завершення отримання поточної форми
createForm	CREATE_FORM	form	Створення форми

Усі модулі оперують загальним інтерфейсом корфігувації форм. За своєю структурою це об’єкт, який містить масив полів у вигляді конфігураційних об’єктів. Структура конфігураційних об’єктів наведена в таблиці 2.4

Таблиці 2.4 - Структура конфігураційних об’єктів

Назва поля	Значення	Призначення
Name	Рядок	задає назву поля
fieldType	‘input’, ‘checkbox’, ‘select’	вказує тип поля
Props	об’єкт (із полем title)	задає статичні властивості поля
State	об’єкт (з полями value, display, disable)	задає динамічні властивості поля

2.3.3 Конструювання серверної частини

Серверна частина реалізується за допомогою платформи Node.js, з використанням серверу на Express.js. Даний підхід дозволяє уніфікувати javascript - як мову загально призначення для спрощення процесу розробки системи автоматизації варіативних опитувань.

Як сховище даних використовується база даних MongoDb. [7]

Даний вибір пояснюється необхідністю зберігати дані у вигляді JSON-формату, з чим дана база даних справляється досить непогано, як зі сторони швидкодії та ефективності, так і зі сторони зручності користування базою.

Структуру вихідних даних можна переглянути на рисунку 2.10

Рисунок 2.10 – Структура вихідних даних

2.4 Аналіз безпеки даних

В якості протоколу взаємодії клієнт-серверної архітектури використовується HTTPS. Протокол працює на основі протоколу HTTP з додатковим функціоналом для підвищення безпеки даних. А саме криптографічні протоколи SSL та TLS.

Для авторизації клієнта використовується технологія JWT.

2.5 Висновки по розділу

Для покращення гнучкості система було прийнято рішення розроблення окремої бібліотеки побудови динамічних форм. Оскільки потрібно вирішити питання автооновлення полів, потрібно було налаштувати їх взаємодію, побудувати процес обміну інформації між ними. Для цього був використаний шаблон проектування “Спостерігач”. Дана бібліотека повинна задовольняти вимогам ефективного оновлення стану в середовищі веб-браузера, де оновлення DOM - одна з найзатратніших задач. Тому для вирішення цієї

проблеми було використано високоефективну бібліотеку побудови користувацького інтерфейсу - React. Для керування станом додатку було впровадження архітектури Flux, за допомогою бібліотеки, яка імплементує цю архітектуру - Redux. На серверній частині також мова програмування javascript, яка виконується на платформі Node.js. Сервер побудований на бібліотеці Express, я роль сховища даних виконує база даних на MongoDb.
Взаємодія між клієнтом та сервером відбувається за допомогою захищеного протоколу HTTPS.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Аналіз якості має велике значення розробці програмних продуктів будь-якої складності, саме це є способом детермінування та формалізації успішності реалізації системи та виконання нею поставлених цілей та вимог.

Ціллю даного тест плану являється опис процесу тестування системи автоматизації варіативних опитувань.

Цільовою аудиторією є особи, які мають труднощі із збором інформації спричинені складністю форм опитувань.

Межі застосування

Тестовий план створений задля тестування ПЗ «Комплекс задач з автоматизації варіативних опитувань» в цілому та її окремих модулів, контролю та планування тестування, передбачення результатів тестування. Сервер має HTTP API, яке базується на JSON, що в комплексі з візуальним інтерфейсом значно полегшує тестування.

3.2 Опис процесів тестування

Усі елементи, з яких складається «Комплекс задач з автоматизації варіативних опитувань» мають бути протестовані. Основні компонентів, що тестуються знаходяться у таблиці 3.1

Таблиця 3.1 - Основні компоненти, що тестуються

Компоненти, що тестуються	Номер версії
Система аутентифікації	v1.0.4
Система авторизації	v1.0.4

Продовження таблиці 3.1

Система персональних даних	v1.0.1
Система створених опитувань	v1.4.5
Підсистема створення опитувань	v1.4.4

Перерахування компонентів, що не тестуються знаходяться у таблиці 3.2

Таблиця 3.2 - Компоненти, що не тестуються

Компоненти, що не тестуються	Номер версії	Причина
Система статистики по пройденим опитуванням	v1.3.4	Вимагає попереднього створення великої кількості опитувань, є другорядним компонентом
Система експорту результатів	v1.3.0	Є другорядним Компонентом

Проектна документація:

- специфікація вимог;
- проектна документація;
- довідник користувача.

Особливості, що тестуються наведені в таблиці 3.3

Таблиця 3.3 - Особливості, що тестуються

Компоненти, що тестуються	Бізнес сценарії, що тестуються
Система аутентифікації	Вхід користувача в систему
Система авторизації	Перевірка відповідності прав користувача
Система персональних даних	Можливість зміни користувацьких даних
Система історії (пройдених опитувань)	Збереження раніше пройдених опитувань
Система створених опитувань	Збереження раніше створених опитувань
Підсистема створення опитувань	Створення опитувань користувачем
Підсистема редагування опитувань	Можливість внесення змін в опитування

Особливості, що не тестуються наведені в таблиці 3.4

Таблиця 3.4 - Особливості, що не тестуються

Компоненти, що не тестуються	Бізнес сценарії, що не тестуються
Система статистики по пройденим опитуванням	Отримання аналітичних даних до пройденим опитуванням
Система експорту результатів	Експорт результатів у вихідні файли

Мета тесту - перевірка функціональності комплексу відповідно до вимог специфікації.

Детальніше опис процесів тестування, критерії проходження тестування та вимоги до середовища можна переглянути в документі “Програма та методика тестування”.

3.3 Опис контрольного прикладу

Опис контрольного прикладу можна переглянути в документі “Програма та методика тестування”.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Для розгортання серверної частини даної системи необхідно мати встановлену платформу Node.js версії 8 та вище. Також для серверної частини необхідно мати встановлену систему управління базами даних MongoDb, для встановлення сторонніх модулів необхідно встановити систему управління пакетами NPM, для підвищення зручності роботи з пакетами можна використати також Yarn, який базується на NPM.

Для запуску клієнтської частини необхідний також встановлений компанувальник проектів Webpack, 3-ої версії та транапілятор Babel. Для запуску проекту в режимі розробки необхідно використати webpack-dev-server.

4.2 Робота з програмним забезпеченням

Детальний опис роботи з програмним забезпеченням наведений в документі “Керівництво користувача”.

Процес розгортання програмного забезпечення досить прозорий та визначений конкретними вимогами. Його можна охарактеризувати, як типовий для даного класу задач, оскільки в розробці використовувався загальноприйнятий стек технологій - MERN, в абревіатурі якого сховані початкові літери назв технологій (MongoDb - система управління базами даних, Express - бібліотека для запуску серверу, React - бібліотека побудови клієнтських інтерфейсів, Node - платформа для запуску JavaScript коду).

ВИСНОВКИ

В ході роботи був виконаний детальний аналіз схожих програмних продуктів, відокремлення їх слабких сторін, та формування функціональних вимог з ціллю реалізації програмного продукту, ключовими характеристиками якого стануть виправлення слабких сторін схожих продуктів. Основною перевагою можна назвати повну керованість процесом побудови форми опитування та як наслідок можливість реалізації дійсно складних та варіативних опитувань.

В процесі роботи було прийнято рішення розробки власної бібліотеку побудови та управління динамічними формами. Це дало змогу ефективного перевикористання логіки побудов форм, а також підвищило гнучкість та можливості системи. Для реалізації проекту було враховано можливі проблеми з швидкодією роботи динамічних форм, для вирішення яких було використано спеціальний стек технологій на чолі з React.

Було проведено аналіз якості програмного забезпечення та наведено опис процесів тестування, для ефективної та детальної реалізації продукту за необхідний час.

Також було наведено опис процесу розгортання програмного забезпечення та його підтримки. Окремо, було реалізовано інструкцію користувача для мінімізації часу, необхідного для ознайомлення з продуктом.

ПЕРЕЛІК ПОСИЛАНЬ

1. Паніна Н. В. Технологія соціологічного дослідження. — К., 1996.
2. Піча В. М., Вовканич С. И., Маковецький В. М. Як підготувати, провести і узагальнити результати соціологічних досліджень? — Львів, 1996.
3. Ядов В. А. Социологическое исследование: методология, программа, методы. — Самара, 1995.
4. Object Management Group Business Process Model and Notation [Електронний ресурс]. – Режим доступу: http://www.bpmn.org/.
5. Design Patterns: Elements of Reusable Object-Oriented Software / E.Gamma, J. Vlissides, R. Johnson, R. Helm., 1994. – 395 с.
6. React - A JavaScript library for building user interfaces [Електронний ресурс]. – Режим доступу: https://reactjs.org/.
7. MongoDB: The most popular database for modern apps [Електронний ресурс]. – Режим доступу: https://www.mongodb.com.
8. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides (1994). Design Patterns: Elements of Reusable Object-Oriented Software.
9. Итан Браун. Веб-разработка с применением Node и Express.
10. Express.js home page [Електронний ресурс]. – Режим доступу: https://expressjs.com/.

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ О.А. Павлов

“___” _____ 2019 р.

КОМПЛЕКС ЗАДАЧ З АВТОМАТИЗАЦІЇ ВАРІАТИВНИХ ОПИТУВАНЬ

Опис програми

КПІ.ІП-5115.045440.02.13

“ПОГОДЖЕНО”

Керівник проекту:

_____О.В. Ковтунець

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____В.Р. Павлюк

Київ – 2019 року

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ О.А. Павлов

“___” _____ 2019 р.

КОМПЛЕКС ЗАДАЧ З АВТОМАТИЗАЦІЇ ВАРІАТИВНИХ ОПИТУВАНЬ

Технічне завдання

КПІ.ІП-5115.045440-02-91

“ПОГОДЖЕНО”

Керівник проекту:

_____ О.В. Ковтунець

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____В.Р. Павлюк

Київ – 2019 року

```
Actions
export const GET_YOUR_FORMS = 'GET_YOUR_FORMS'
export const getYourForms = (userId) => ({
 type: GET_YOUR_FORMS,
 userId,
})

export const GOT_YOUR_FORMS = 'GOT_YOUR_FORMS'
export const gotYourForms = (data) => ({
 type: GOT_YOUR_FORMS,
 data,
})

export const GET_CURRENT_FORM = 'GET_CURRENT_FORM'
export const getCurrentForm = (id, userId) => ({
 type: GET_CURRENT_FORM,
 id,
 userId,
})

export const GOT_CURRENT_FORM = 'GOT_CURRENT_FORM'
export const gotCurrentForm = (data) => ({
 type: GOT_CURRENT_FORM,
 data,
})

export const CREATE_FORM = 'CREATE_FORM'
export const createForm = (form, userId) => ({
 type: CREATE_FORM,
```

```
 form,
 userId,
})

export const GOT_CREATED_FORM = 'GOT_CREATED_FORM'
export const gotCreatedForm = (form) => ({
 type: GOT_CREATED_FORM,
 form,
})

export const UPDATE_FORM = 'UPDATE_FORM'
export const updateForm = (form, id, userId) => ({
 type: UPDATE_FORM,
 form,
 id,
 userId,
})

export const GOT_UPDATED_FORM = 'GOT_UPDATED_FORM'
export const gotUpdatedForm = (form) => ({
 type: GOT_UPDATED_FORM,
 form,
})

// answers
export const GET_FORM_ANSWERS = 'GET_FORM_ANSWERS'
export const getFormAnswers = (formId) => ({
 type: GET_FORM_ANSWERS,
 formId,
```

```
})

export const GOT_FORM_ANSWERS = 'GOT_FORM_ANSWERS'
export const gotFormAnswers = (answers) => ({
 type: GOT_FORM_ANSWERS,
 answers,
})

export const SEND_FORM_ANSWERS = 'SEND_FORM_ANSWERS'
export const sendFormAnswers = (formId, answer, userId) => ({
 type: SEND_FORM_ANSWERS,
 formId,
 answer,
 userId,
})

export const FORM_ANSWERS_SENT = 'FORM_ANSWERS_SENT'
export const formAnswersSent = (status) => ({
 type: FORM_ANSWERS_SENT,
 status,
})


export const GET_USER = 'GET_USER'
export const getUser = (id) => ({
 type: GET_USER,
 id,
})

export const GOT_USER = 'GOT_USER'
```

```
export const gotUser = (data) => ({
 type: GOT_USER,
 data,
})

export const CREATE_USER = 'CREATE_USER'
export const createUser = (user) => ({
 type: CREATE_USER,
 user,
})

export const GOT_CREATED_USER = 'GOT_CREATED_USER'
export const gotCreatedUser = (user) => ({
 type: GOT_CREATED_USER,
 user,
})

export const USER_AUTH = 'USER_AUTH'
export const userAuth = (user) => ({
 type: USER_AUTH,
 user,
})

export const USER_AUTHED = 'USER_AUTHED'
export const userAuthed = (user) => ({
 type: USER_AUTHED,
 user,
})
```

```
export const USER_NOT_EXIST = 'USER_NOT_EXIST'
export const userNotExist = () => ({
 type: USER_NOT_EXIST,
})

export { simpleAction } from './simpleAction'
export { incrementAction } from './incrementAsync'

export { GET_USER, getUser } from './requestActions'
export { GOT_USER, gotUser } from './requestActions'
export { CREATE_USER, createUser } from './requestActions'
export { GOT_FAILURE, gotFailure } from './requestActions'

export const LOG_OUT = 'LOG_OUT'
export const logOutAction = () => ({
 type: LOG_OUT,
})
```

Config

```
import {
 GET_YOUR_FORMS, gotYourForms,
 GET_CURRENT_FORM, gotCurrentForm,
 CREATE_FORM, gotCreatedForm,
 UPDATE_FORM, gotUpdatedForm,
 GET_FORM_ANSWERS, gotFormAnswers,
 SEND_FORM_ANSWERS, formAnswersSent,
 CREATE_USER, gotCreatedUser,
 userNotExist, USER_AUTH, userAuthed,
```

```
 gotFailure,
} from '../actions/requestActions'

export const host = 'http://localhost:3000'
export const frontendHost = 'http://localhost:8080/#'

const apiConfig = {
 getYourForms: {
   triggerActionType: GET_YOUR_FORMS,
   successAction: gotYourForms,
   failureAction: gotFailure,
   getOptions: ({ userId }) => ({
     method: 'post',
     url: `${host}/forms`,
     data: { userId },
   }),
 },
 getCurrentForm: {
   triggerActionType: GET_CURRENT_FORM,
   successAction: gotCurrentForm,
   failureAction: gotFailure,
   getOptions: ({ id, userId }) => ({
     method: 'post',
     url: `${host}/formById/${id}`,
     data: { userId },
   }),
 },
 createForm: {
   triggerActionType: CREATE_FORM,
```

```
  successAction: gotCreatedForm,
  failureAction: gotFailure,
  getOptions: ({ form, userId }) => ({
    method: 'post',
    url: `${host}/createForm`,
    data: { form, userId },
  }),
},
updateForm: {
  triggerActionType: UPDATE_FORM,
  successAction: gotUpdatedForm,
  failureAction: gotFailure,
  getOptions: ({ id, form, userId }) => ({
    method: 'post',
    url: `${host}/updateForm/${id}`,
    data: { form, userId },
  }),
},
getFormAnswers: {
  triggerActionType: GET_FORM_ANSWERS,
  successAction: gotFormAnswers,
  failureAction: gotFailure,
  getOptions: ({ formId }) => ({
    method: 'get',
    url: `${host}/answers/${formId}`,
  }),
},
sendFormAnswers: {
  triggerActionType: SEND_FORM_ANSWERS,
```

```
    successAction: formAnswersSent,
    failureAction: gotFailure,
    getOptions: ({ formId, answer, userId }) => ({
      method: 'post',
      url: `${host}/answers`,
      data: { formId, answer, userId },
    }),
  },
  createUser: {
    triggerActionType: CREATE_USER,
    successAction: gotCreatedUser,
    failureAction: gotFailure,
    getOptions: ({ user }) => ({
      method: 'post',
      url: `${host}/users`,
      data: { user },
    }),
  },
  urerAuth: {
    triggerActionType: USER_AUTH,
    successAction: userAuthed,
    failureAction: userNotExist,
    getOptions: ({ user }) => ({
      method: 'post',
      url: `${host}/auth`,
      data: { user },
    }),
  },
}
```

```
export default apiConfig

import apiConfig, { host, frontendHost } from './api'

const config = {
 api: apiConfig,
 host,
 frontendHost,
}

export default config
```

Result modal

```
import React, { Component } from 'react'
import { connect } from 'react-redux'
import { Container } from 'semantic-ui-react'
import LiveFormBuilder from '../live-form-builder'
import { createForm } from '../../actions/requestActions'
import ModalExampleControlled from './result-modal'

class FormCreator extends Component {
  render() {
    const { createNewForm } = this.props
    const userId = localStorage.getItem('userId')
    return (
      <div>
        <Container>
```

```
        <LiveFormBuilder
          createNewForm={(form) => createNewForm(form, userId)}
        />
        <ModalExampleControlled />
      </Container>
    </div>
  )
 }
}

const mapStateToProps = () => null

const mapDispatchToProps = (dispatch) => ({
  createNewForm: (form, userId) => dispatch(createForm(form, userId)),
})

export default connect(
  mapStateToProps,
  mapDispatchToProps,
)(FormCreator)

FormFiller

import React, { Component } from 'react'
import { withRouter } from 'react-router-dom'
import LiveForm from 'react-live-form'
import { connect } from 'react-redux'
import { Header } from 'semantic-ui-react'
import ResultModal from './result-modal'
```

```
import {
 getCurrentForm,
 sendFormAnswers,
} from '../../actions/requestActions'
import './index.scss'

class FormFiller extends Component {
 componentDidMount() {
   const { getForm, match = {} } = this.props
   const { id } = match.params
   const userId = localStorage.getItem('userId')
   if (getForm) {
     getForm(id, userId)
   }
 }

 onSendAnswers = (answer) => {
   const { sendAnswer, match = {} } = this.props
   const { id } = match.params
   const userId = localStorage.getItem('userId')
   if (sendAnswer) {
     sendAnswer(+id, answer, userId)
   }
 }

 render() {
   const {
     match = {},
     preloadedForm,
```

```
  } = this.props
  const { id } = match.params

  const { title, description } = (preloadedForm || {})

  console.log('id', id)
  console.log('preloadedForm', preloadedForm)

  return (
    <div className="form-filler">
      <div className="header">
        <Header as='h2'>{title}</Header>
        <Header as='h3'>{description}</Header>
      </div>
      {
        preloadedForm && (
          <LiveForm
            onSubmit={this.onSendAnswers}
            formConfig={preloadedForm.formConfig}
          />
        )
      }
      <ResultModal />
    </div>
  )
 }
}

const mapStateToProps = (state) => ({
```

```
 preloadedForm: state.forms && state.forms.currentForm,
})

const mapDispatchToProps = (dispatch) => ({
 getForm: (id, userId) => dispatch(getCurrentForm(id, userId)),
 sendAnswer: (id, answer, userId) => dispatch(sendFormAnswers(id, answer,
userId)),
})

export default connect(
 mapStateToProps,
 mapDispatchToProps,
)(withRouter(FormFiller))
```

FormCreator

```
import React, { Component } from 'react'
import { connect } from 'react-redux'
import { Container } from 'semantic-ui-react'
import LiveFormBuilder from '../live-form-builder'
import { createForm } from '../../actions/requestActions'
import ModalExampleControlled from './result-modal'

class FormCreator extends Component {
 render() {
   const { createNewForm } = this.props
   const userId = localStorage.getItem('userId')
   return (
     <div>
```

```
      <Container>
        <LiveFormBuilder
          createNewForm={(form) => createNewForm(form, userId)}
        />
        <ModalExampleControlled />
      </Container>
    </div>
  )
 }
}

const mapStateToProps = () => null

const mapDispatchToProps = (dispatch) => ({
 createNewForm: (form, userId) => dispatch(createForm(form, userId)),
})

export default connect(
 mapStateToProps,
 mapDispatchToProps,
)(FormCreator)

LiveFormBuilder

import React, { Component } from 'react'
import {
 Divider,
 Grid,
 Segment,
```

```
} from 'semantic-ui-react'
import LiveForm from 'react-live-form'
import { set } from 'utils'
import FormProps from './form-props'
import FieldsController from './fields-controller'
import FieldEditor from './field-editor'

const getInitialState = (props) => {
 const { preloadedForm = {} } = props
 const initialFormConfig = {
   fields: [],
 }

 const formConfig = (preloadedForm && preloadedForm.formConfig) ||
initialFormConfig
 const activeFieldIndex = formConfig.fields.length > 0 ? 0 : null
 const loaded = activeFieldIndex !== null

 return {
   formConfig,
   title: preloadedForm.title || '',
   description: preloadedForm.description || '',
   activeFieldIndex,
   loaded,
 }
}

class LiveFormBuilder extends Component {
 state = getInitialState(this.props)
```

```
static getDerivedStateFromProps(props, state) {
  // Any time the current user changes,
  // Reset any parts of state that are tied to that user.
  // In this simple example, that's just the email.
  if (
    props.preloadedForm &&
    props.preloadedForm.formConfig !== state.formConfig &&
    !state.loaded
  ) {
    return {
      formConfig: props.preloadedForm.formConfig,
      title: props.preloadedForm.title || '',
      description: props.preloadedForm.description || '',
      activeFieldIndex: 0,
      loaded: true,
    }
  }
  return null
}

onChangeFormProps = (prop, value) => {
  this.setState(() => {
    return {
      [prop]: value,
    }
  })
}
```

```
onChangeFormConfig = (newFormConfig) => {
  this.setState(() => {
    return {
      formConfig: newFormConfig,
    }
  })
}

addField = () => {
  this.setState((prevState) => {
    const formConfig = Object.assign({}, prevState.formConfig)
    const currentIndex = prevState.formConfig.fields.length
    const fields = [
      ...prevState.formConfig.fields,
      {
        name: `name${currentIndex}`,
        fieldType: 'input',
        props: {
          title: `Title ${currentIndex}: `,
        },
        state: {
          value: {
            defaultValue: '',
            valueExpr: '',
          },
          display: {
            defaultValue: true,
            valueExpr: '',
          },
```

```
        disabled: {
          defaultValue: false,
          valueExpr: '',
        },
      },
    },
  ]

  formConfig.fields = fields

  return {
    formConfig,
    activeFieldIndex: fields.length - 1,
  }
 })
}

removeField = () => {
 this.setState((prevState) => {
  const { activeFieldIndex, formConfig } = prevState
  const newFormConfig = Object.assign({}, formConfig)
  const fields = [...formConfig.fields]
  fields.splice(activeFieldIndex, 1)
  newFormConfig.fields = fields
  return {
    formConfig: newFormConfig,
    activeFieldIndex: (activeFieldIndex !== 0)
      ? activeFieldIndex - 1
      : null,
```

```
    }
  })
}

onChangeFieldConfig = (key, value) => {
  const { activeFieldIndex, formConfig } = this.state

  const { fields } = formConfig
  const newFields = [...fields]

  const fieldConfig = set(newFields[activeFieldIndex], key, value)
  newFields[activeFieldIndex] = fieldConfig

  const newFormConfig = {
    ...formConfig,
    fields: newFields,
  }

  this.setState(() => {
    return {
      formConfig: newFormConfig,
    }
  })

  console.log('onChangeFieldConfig', activeFieldIndex, key, value)
}

onChangeActiveFieldIndex = (newActiveFieldIndex) => {
  this.setState(() => {
```

```
    return {
      activeFieldIndex: newActiveFieldIndex,
    }
  })
}

getFieldsOptions = () => {
  const { formConfig: { fields } } = this.state
  const options = fields.map((fieldConfig) => {
    const { name } = fieldConfig
    return {
      key: name,
      text: name,
      value: name,
    }
  })
  return options
}

onFormPublish = () => {
  const { createNewForm } = this.props
  const { formConfig, title, description } = this.state

  createNewForm({
    formConfig,
    title,
    description,
  })
}
```

```
render() {
  const {
    activeFieldIndex,
    formConfig,
    title,
    description,
  } = this.state
  const fieldsOptions = this.getFieldsOptions()
  const activeField = formConfig.fields[activeFieldIndex] || {}

  return (
    <div>
      <FormProps
        title={title}
        description={description}
        onChangeFormProps={this.onChangeFormProps}
        onFormPublish={this.onFormPublish}
      />
      <Segment>
        <Grid columns={2} relaxed='very'>
          <Grid.Column>
            <FieldsController
              addField={this.addField}
              removeField={this.removeField}
              onChangeActiveFieldIndex={this.onChangeActiveFieldIndex}
              fieldsOptions={fieldsOptions}
              activeFieldIndex={activeFieldIndex}
            />
```

```
            {
              (activeFieldIndex !== null) &&
              (
                <FieldEditor
                  formConfig={formConfig}
                  activeField={activeField}
                  onChangeFormConfig={this.onChangeFormConfig}
                  onChangeFieldConfig={this.onChangeFieldConfig}
                />
              )
            }
          </Grid.Column>
          <Grid.Column>
            <LiveForm formConfig={formConfig} />
          </Grid.Column>
        </Grid>
        <Divider vertical>preView</Divider>
      </Segment>
    </div>
  )
 }
}

export default LiveFormBuilder

FieldEditor

import React, { Component } from 'react'
import { Form, TextArea } from 'semantic-ui-react'
```

```
import FieldMainInputs from './field-main-inputs'
import FieldPropsInputs from './field-props-inputs'
import FieldStateInputs from './field-state-inputs'

class FieldEditor extends Component {
 render() {
  const {
    activeField,
    formConfig,
    onChangeFormConfig,
    onChangeFieldConfig,
  } = this.props
  return (
    <div>
      <Form>
        <FieldMainInputs
          activeField={activeField}
          onChangeFieldConfig={onChangeFieldConfig}
        />
        <FieldPropsInputs
          activeField={activeField}
          onChangeFieldConfig={onChangeFieldConfig}
        />
        <FieldStateInputs
          activeField={activeField}
          onChangeFieldConfig={onChangeFieldConfig}
        />
        <Form.Field
          id="form-textarea-control-opinion"
```

```
          control={TextArea}
          label="Opinion"
          placeholder="Opinion"
          rows={7}
          value={JSON.stringify(formConfig, null, 2)}
          onChange={(e, d) => onChangeFormConfig(JSON.parse(d.value))}
        />
        {/*
        <Form.Field
          id="form-button-control-public"
          control={Button}
          content="Confirm"
          label='Label with htmlFor'
        /> */}
      </Form>
    </div>
  )
 }
}

export default FieldEditor

FieldsInputProps

import React, { Component } from 'react'
import { Form } from 'semantic-ui-react'
import { get } from 'utils'

class FieldPropsInputs extends Component {
```

```
addSelectFieldOption = () => {
  const {
    activeField,
    onChangeFieldConfig,
  } = this.props

  const options = get(activeField, 'props.options') || []
  const newOptions = [...options]
  const index = newOptions.length + 1

  newOptions.push({
    value: `option${index}`,
    content: `Option${index}`,
  })

  onChangeFieldConfig('props.options', newOptions)
}

removeSelectFieldOption = (index) => {
  const {
    activeField,
    onChangeFieldConfig,
  } = this.props

  const options = get(activeField, 'props.options') || []
  const newOptions = [...options]
  newOptions.splice(index, 1)

  onChangeFieldConfig('props.options', newOptions)
```

```
}

getSelectOptionsInputs = () => {
  const {
    activeField,
    onChangeFieldConfig,
  } = this.props

  const {
    fieldType,
    props: { options = [] },
  } = activeField

  let selectOptionsInputs = null
  if (fieldType === 'select') {
    const optionsInputs = Array.from(options).map((option = {}, index) => {
      const { value, content } = option
      return (
        <div key={index}>
          <Form.Group widths="equal">
            <Form.Input
              fluid
              label="Option value"
              placeholder="Option"
              value={value}
              onChange={
                (e, d) => onChangeFieldConfig(`props.options.${index}.value`, d.value)
              }
            />
```

```
        <Form.Input
          fluid
          label="Option content"
          placeholder="Content"
          value={content}
          onChange={
            (e, d) => onChangeFieldConfig(`props.options.${index}.content`, d.value)
          }
        />
        <Form.Button
          fluid
          negative
          label="Remove option"
          onClick={() => this.removeSelectFieldOption(index)}
        >
          Remove
        </Form.Button>
      </Form.Group>
    </div>
  )
})

selectOptionsInputs = (
  <>
    <Form.Group widths="equal">
      <Form.Button
        positive
        onClick={this.addSelectFieldOption}
      >
```

```
            Add Option
          </Form.Button>
        </Form.Group>
        {optionsInputs}
      </>
    )
  }

  return selectOptionsInputs
}

render() {
  const { activeField, onChangeFieldConfig } = this.props
  const selectOptionsInputs = this.getSelectOptionsInputs()

  return (
    <>
      <h3>Field props</h3>
      <Form.Input
        fluid
        label="Field title"
        placeholder="Title"
        value={get(activeField, 'props.title')}
        onChange={(e, d) => onChangeFieldConfig('props.title', d.value)}
      />
      {selectOptionsInputs}
    </>
  )
}
```

```
}

export default FieldPropsInputs
```

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ О.А. Павлов

“___” _____ 2019 р.

КОМПЛЕКС ЗАДАЧ З АВТОМАТИЗАЦІЇ ВАРІАТИВНИХ ОПИТУВАНЬ

Технічне завдання

КПІ.ІП-5115.045440.03.91

“ПОГОДЖЕНО”

Керівник проекту:

_____О.В. Ковтунець

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____В.Р. Павлюк

Київ – 2019 року

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ О.А. Павлов

“___” _____ 2019 р.

КОМПЛЕКС ЗАДАЧ З АВТОМАТИЗАЦІЇ ВАРІАТИВНИХ ОПИТУВАНЬ

Технічне завдання

КПІ.ІП-5115.045440.03.91

“ПОГОДЖЕНО”

Керівник проекту:

_____ О.В. Ковтунець

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____В.Р. Павлюк

Київ – 2019 року

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки:

Галузь застосування:

Наведене технічне завдання поширюється на розробку програмного забезпечення “Комплекс задач з автоматизації варіативних опитувань” , котра використовується для полегшення створення складних опитувань з логічно зв’язаними блоками питань, та призначена для галузей, в роботі яких існує необхідність створення складних, логічно структурованих опитувань, де можливі взаємозв’язки між полями. Типовим прикладом є страхові компанії, в робочому циклі яких існує необхідність опитування клієнтів, медичні заклади, які можуть впроваджувати систему попереднього опитування пацієнтів у електронному вигляді для економії часу та фізичних носіїв (бланків). Також як галузь застосування можна розглянути відділ продаж більшості компаній, які в процесі взаємодії з клієнтом використовують систему скриптів, де відповіді на поточні питання змінюють наступні або переходять до уточнюючих запитань.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки “Комплексу задач з автоматизації варіативних опитувань” є завдання на дипломне проектування, затверджене кафедрою автоматизованих систем обробки інформації і управління Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім.Ігоря Сікорського).

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для полегшення створення складних опитувань з логічно зв’язаними блоками питань.

Метою розробки є полегшення процесу створення опитувань з логічно зв’язаними структурними блоками. Підвищення зручності їх використання як зі сторони адміністратора, та і зі сторони користувача за допомогою розробки одно-сторінкового веб-додатку, із збереженням даних на серверній частині.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

4.1.1 Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

4.1.1.1 Для користувача:

- зареєструватися;
- авторизуватися, для зареєствованих користувачів;
- перглянути опитування;
- заповнити опитування;
- надіслати результати опитування.

4.1.1.2 Для адміністратора системи:

- авторизуватися в системі;
- створювати опитування;
- оновлювати опитування;
- видалити опитування;
- переглядати результати опитувань;
- надіслати створене опитування користувачу.

4.1.2 Розробку виконати у вигляді SPA додатку

4.1.3 Додаткові вимоги:

- підтримка усіма сучасними браузерами (2 останні версії);
- підтримка усіма розширеннями екранів;
- передача даних за допомогою REST архітектури;
- розмір бандлу не перевищує 1мб javascript коду.

4.2 Вимоги до надійності

4.2.1 Передбачити контроль введення інформації.

4.2.2 Передбачити захист від некоректних дій користувача.

4.2.3 Забезпечити цілісність інформації в базі даних.

4.3 Умови експлуатації

4.3.1 Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.4 Вимоги до складу і параметрів технічних засобів

4.4.1 Програмне забезпечення повинно функціонувати на IBM-сумісних персональних комп‘ютерах.

4.4.2 Мінімальна конфігурація технічних засобів:

Тип процесору Pentium.
Об‘єм ОЗП 256 Мб.

4.5 Вимоги до інформаційної та програмної сумісності

4.5.1 Програмне забезпечення повинно працювати під управлінням операційних систем сімейства WIN32 (Windows'XP, Windows NT і т.д.) або Unix.

4.5.2 Вхідні дані повинні бути представлені в JSON форматі

Дана структура повинна включати назву форми опитування, її короткий опис та набір полів, з яких складається дана форма.

Приклад набору полів:

```
const formConfig = {
  fields: [
    {
```

```
  name: 'a',
  fieldType: 'input',
  dataType: 'int',
  props: {
    title: 'field a',
  },
  state: {
    value: {
      defaultValue: 5,
    },
  },
},
{
  name: 'b',
  fieldType: 'input',
  dataType: 'int',
  props: {
    title: 'field b',
  },
},
{
  name: 'c',
  fieldType: 'input',
  dataType: 'string',
  props: {
    title: 'field c',
  },
  state: {
    value: {
      defaultValue: 0,
      valueExpr: 'a + b',
    },
  },
},
{
  name: 'd',
  fieldType: 'input',
```

```
      dataType: 'string',
      props: {
        title: 'field d',
      },
      state: {
        value: {
          defaultValue: 0,
          valueExpr: 'c * 2',
        },
        display: {
          defaultValue: false,
          valueExpr: 'a > 10',
        },
        disabled: {
          defaultValue: false,
          valueExpr: 'a > 34',
        },
      },
    },
  ],
}
```

4.5.3 Результати повинні бути представлені JSON-форматі, яких за своєю структурою складається з масиву об’єктів, які відповідають об’єктам вхідних даних.

4.6 Вимоги до мови і середовища розробки

Даний комплекс повинен бути розробленим із застосуванням мови програмування JavaScript, використовуючи версію, яка відповідає специфікації EcmaScript6.

Для розробки клієнтської частини слід використати бібліотеку побудови інтерактивних інтерфейсів основану на ідеологогії віртуального DOM – React, версії 16+. Для керування станом додатку використати бібліотеку Redux, для обробки асинхроних подій Redux-Saga.

Для серверної частини використати платформу Node.js з сервером на Express.

4.7 Вимоги до маркування та пакування

Вимоги до маркування та пакування не пред‘являються.

4.8 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не пред'являються.

4.9 Спеціальні вимоги

Згенерувати установчу версію програмного забезпечення.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

5.2 Програмне забезпечення повинно мати довідникову систему

5.3 У склад супроводжувальної документації повинні входити наступні документи:

5.3.1 Пояснювальна записка не менше ніж на 60 аркушах формату А4 (без додатків 5.3.2 - 5.3.6).

5.3.2 Технічне завдання.

5.3.3 Керівництво користувача.

5.3.4 Керівництво адміністратора

5.3.5 Програма та методика тестування

5.4 Графічна частина повинна бути виконана на листі формату А3, котрі включаються у якості додатків до пояснювальної записки:

5.4.1 Схема структурна варіантів використання

5.4.2 Схема структурна класів програмного забезпечення

5.4.3 Креслення вигляду екранних форм.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

	Назва етапу	Строк,	Звітність
1	Вивчення літератури за тематикою проекту	20.04.19	
2	Розробка технічного завдання	23.04.19	Технічне завдання
3	Аналіз вимог та уточнення специфікацій	25.04.19	Специфікації програмного забезпечення
4	Проектування структури програмного забезпечення, проектування компонентів	28.04.19	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму …)
5	Програмна реалізація програмного забезпечення	15.05.19	Тексти програмного забезпечення
6	Тестування програмного забезпечення	19.05.19	Тести, результати тестування
7	Розробка матеріалів текстової частини проекту	27.05.19	Пояснювальна записка.
8	Розробка матеріалів графічної частини проекту	07.06.19	Графічний матеріал проекту

9	Оформлення технічної документації проекту	07.06.19	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

7.1 Види випробувань

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ О.А. Павлов

“___” _____ 2019 р.

КОМПЛЕКС ЗАДАЧ З АВТОМАТИЗАЦІЇ ВАРІАТИВНИХ ОПИТУВАНЬ

Програма та методика тестування

КПІ.ІП-5115.045440.04.51

“ПОГОДЖЕНО”

Керівник проекту:

_____О.В. Ковтунець

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____В.Р. Павлюк

Київ – 2019 року

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ О.А. Павлов

“___” _____ 2019 р.

КОМПЛЕКС ЗАДАЧ З АВТОМАТИЗАЦІЇ ВАРІАТИВНИХ ОПИТУВАНЬ

Програма та методика тестування

КПІ.ІП-5115.045440.04.51

“ПОГОДЖЕНО”

Керівник проекту:

_____О.В. Ковтунець

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____ В.Р. Павлюк

Київ – 2019 року

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Усі складові SPA-додатку, який представляє собою “Комплекс задач з автоматизації варіативних опитувань” мають бути протестовані на відповідність заданим вимогам.

Основні функціональні вимоги додатку, що мають бути протестовані:

- реєстрація;
- авторизація;
- створення опитування;
- налаштування автоматичної модифікації стану полів:
  - за допомогою логічних операторів;
  - за допомогою математичних операторів;
- перегляд створених опитувань;
- оновлення опитування;
- заповнення опитування;
- надсилання результатів опитування;
- перегляд результатів опитування.

2 МЕТОДИ ТЕСТУВАННЯ

Тестування за знанням системи виконується методом сірої скриньки.

Так, як даний підхід є комбінацією методів білої скриньки, коли у нас є доступ до реалізації системи, та чорної скриньки, який позиціонує систему з точки зору користувача, у нас з'являється змога комплексно перевірити працездатність програмного забезпечення, а також відповідність його реалізації поставленим вимогам.

Частина тестів буде виконуватися у ручному режимі, інша автоматично. Автоматично будуть виконуватися модульні та інтеграційні тести. Системні тести будуть виконуватися у ручному режимі.

2.1 Модульне тестування

Модульне тестування призначене для перевірки коректності роботи окремого модуля, причому під час проходження тестів модуль повністю ізолюється від зовнішніх факторів. Усі процеси, пов'язані з введенням та виведенням інформації, а також взаємодія із іншими модулями емалюється. Такий підхід дає змогу ефективно перевірити робото здатність конкретного модуля, а також пришвидшує проходження тестування.

Модульними тестами покриті усі функції побудованої бібліотеки react-live-forms (виконує основну логіку керування динамічними формами додатку), так як вона розроблялася за принципом TTD, дана методологія затверджує процес розробки через тестування. Суть полягає в тому, що спочатку розробляється модульний тест, описуються усі можливі сценарії його поведінки і лише після цього настає етап реалізації модуля, який повинен виконувати усі сценарії даного тесту. Таким чином збільшується надійність програмного забезпечення, а покриття тестами досягає максимального значення.

2.2 Системне тестування

Тестування системи на найвищому рівні відбувається за допомогою системного підходу. Системне тестування на відмінну від інтеграційного та модульного перевіряє коректність роботи систему у цілому. Перевіряється взаємодія усіх складових системи методом чорної скриньки, тобто зі сторони користувача (на рівні тестування інтерфейсу). Ціллю даного типу тестування є виявлення невідповідності роботи системи заявленим вимогам.

За допомогою системного тестування перевіряється працездатність наступних функціональних вимог:

- реєстрація адміністратора;
- авторизація адміністратора;
- створення опитування адміністратором;
- модифікація опитування адміністратором;
- заповнення опитування користувачем;
- перегляд результатів опитування адміністратором.

3 ПРОЦЕС ТЕСТУВАННЯ

3.1 Автоматизоване тестування

Після кожного оновлення кодової бази система автоматично буде запускати, за допомогою сервісів безперервної інтеграції, лінтер на перевірку відповідності коду заданим стандартам, та для виявлення синтаксичних помилок, після цього будуть запускатися модульні тести, які пов’язані з модулями в яких були правки (для пришвидшення проходження тестів на даному етапі). У випадку розгортання на середовищі розробки автоматично запускаються модульні тести, які виконують роль димного тестування (перевіряються головні системи та підсистеми сервісу):

- реєстрація;
    - перевірка створення нового користувача;
- авторизація;
    - авторизація за допомогою даних новоствореного користувача;
- підсистема створення опитувань;
    - створення новим користувачем опитування;
- підсистема редагування опитувань;
    - редагування створеного опитування;
- система проходження опитування;
    - реєстрація, ще одного користувача;
    - його авторизація;
    - проходження опитування;

Також можливий запуск тестування конкретних систем (не автоматичний).

Залежності системи, що тестується за рівнем тестування від часу наведені в таблиці 3.1

Таблиця 3.1 - Залежність системи, що тестується від часу

Особливість, що стосується	Рівень Тестування	Час тестування	Інструмент
Система реєстрації	Модульне тестування	Оновлення кодової бази	jest/enzyme
	Інтеграційне тестування	Нове розсортування	ci/cd, jest/enzyme, protractor
	Приймальне тестування	Випуск нової версії	
Система авторизації	Модульне тестування	Оновлення кодової бази	jest/enzyme
	Інтеграційне тестування	Нове розгортування	ci/cd, jest/enzyme, protractor
	Приймальне тестування	Випуск нової версії	
Підсистема створення опитувань	Модульне тестування	Оновлення кодової бази	jest/enzyme
	Інтеграційне тестування	Нове розгортування	ci/cd, jest/enzyme, protractor
	Приймальне тестування	Випуск нової версії	

Продовження таблиці 3.1

Підсистема редагування опитувань	Модульне тестування	Оновлення кодової бази	jest/enzyme
	Інтеграційне тестування	Нове розсортування	ci/cd, jest/enzyme, protractor
	Приймальне тестування	Випуск нової версії	
Система історія (пройдених опитувань)	Модульне тестування	Оновлення кодової бази	jest/enzyme

3.2 Ціль тестування

Ціллю тестування є перевірка роботи системи відповідно з заявленими вимогами до цієї системи. Виявлення дефектів роботи на різних етапах розробки.

4 ВИМОГИ ДО СЕРЕДОВИЩА

4.1 Модульне тестування

Апаратне забезпечення:

- персональний комп’ютер.

Програмне забезпечення:

- встановлене середовище розробки Atom;
- встановлена платформа Node.js;
- середовище управління тестами Jest;
- засіб ізоляції функцій Sinon.

4.2 Інтеграційне тестування

Апаратне забезпечення:

- персональний комп’ютер.

Програмне забезпечення:

- встановлене середовище розробки Atom;
- встановлена платформа Node.js;
- середовище управління тестами Jest;
- бібліотека порівняння Chai;

4.3 тестування інтерфейсу:

Апаратне забезпечення:

- персональний комп’ютер.

Програмне забезпечення:

- встановлене середовище розробки Atom;
- встановлена платформа Node.js;
- браузер на основі Webkit.

4.4 Тестування usability:

Апаратне забезпечення:

- персональний комп’ютер.

Програмне забезпечення:

- встановлене середовище розробки Atom;
- встановлена платформа Node.js;
- protractor;
- браузер на основі Webkit.

5 ОПИС КОНТРОЛЬНОГО ПРИКЛАДУ

Таблиця 5.1 – Реєстрація користувача

Мета тесту	Перевірка реєстрації користувача
Початковий стан	Користувач зайшов в додаток
Вхідні дані	Ім’я, прізвище, пошта, пароль користувача
Схема проведення тесту	При вході в систему після автоматичного перенаправлення на сторінку авторизації необхідно перейти на сторінку реєстрації, де необхідно ввести дані ім’я, прізвища, пошти, паролю. Після чого підтвердити пароль ще раз та натиснути на кнопку “далі”. Після цього користувач має відбутися реєстрація нового користувача
Очікуваний результат	Користувач зареєструвався в систему, автоматичне пере направлення на сторінку авторизації

Таблиця 5.2 - Створення опитування

Мета тесту	Перевірка створення опитування
Початковий стан	Адміністратор пройшов авторизацію
Вхідні дані	Назва, опис, структура опитування
Схема проведення тесту	Адміністратор з особистого кабінету переходить до конструктора опитування, де створюючи поля одне за одним формує форму. Далі необхідно заповнити поля з назвою та описом опитування, після чого натиснути кнопку ‘створити’.
Очікуваний результат	Повідомлення про створення нового опитування

Таблиця 5.3 - Проходження опитування

Мета тесту	Перевірка проходження опитування
Початковий стан	Користувачу було надіслане опитування
Вхідні дані	Відповіді на поля опитування
Схема проведення тесту	Користувачу необхідно перейти по надісланому посиланню, після чого користувач отримує форму на заповнення. Після заповнення усіх полів форми та потрібно натиснути кнопку “далі” для надсилання відповідей адміністратору форми
Очікуваний результат	Повідомлення про надіслення відповідей

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ О.А. Павлов

“___” _____ 2019 р.

КОМПЛЕКС ЗАДАЧ З АВТОМАТИЗАЦІЇ ВАРІАТИВНИХ ОПИТУВАНЬ

Керівництво користувача

КПІ.ІП-5115.045440.05.34

“ПОГОДЖЕНО”

Керівник проекту:

_____О.В. Ковтунець

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____В.Р. Павлюк

Київ – 2019 року

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ О.А. Павлов

“___” _____ 2019 р.

КОМПЛЕКС ЗАДАЧ З АВТОМАТИЗАЦІЇ ВАРІАТИВНИХ ОПИТУВАНЬ

Керівництво користувача

КПІ.ІП-5115.045440.05.34

“ПОГОДЖЕНО”

Керівник проекту:

_____О.В. Ковтунець

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____ В.Р. Павлюк

Київ – 2019 року

ї

ЗМІСТ

1 ІНСТРУКЦІЯ КОРИСТУВАЧА

1.1 Реєстрація користувача

Для реєстрації користувача необхідно перейти на сторінку реєстрації по роуту ‘/sign-up’. Далі потрібно ввести контактну інформацію в поля вводу. Необхідно ввести своє ім’я, прізвище, електронну пошту та пароль, після чого підтвердити погодження з правилами використання контактної інформації користувача та натиснути кнопку відправки. Сторінка реєстрації зображена на рисунку 1.1

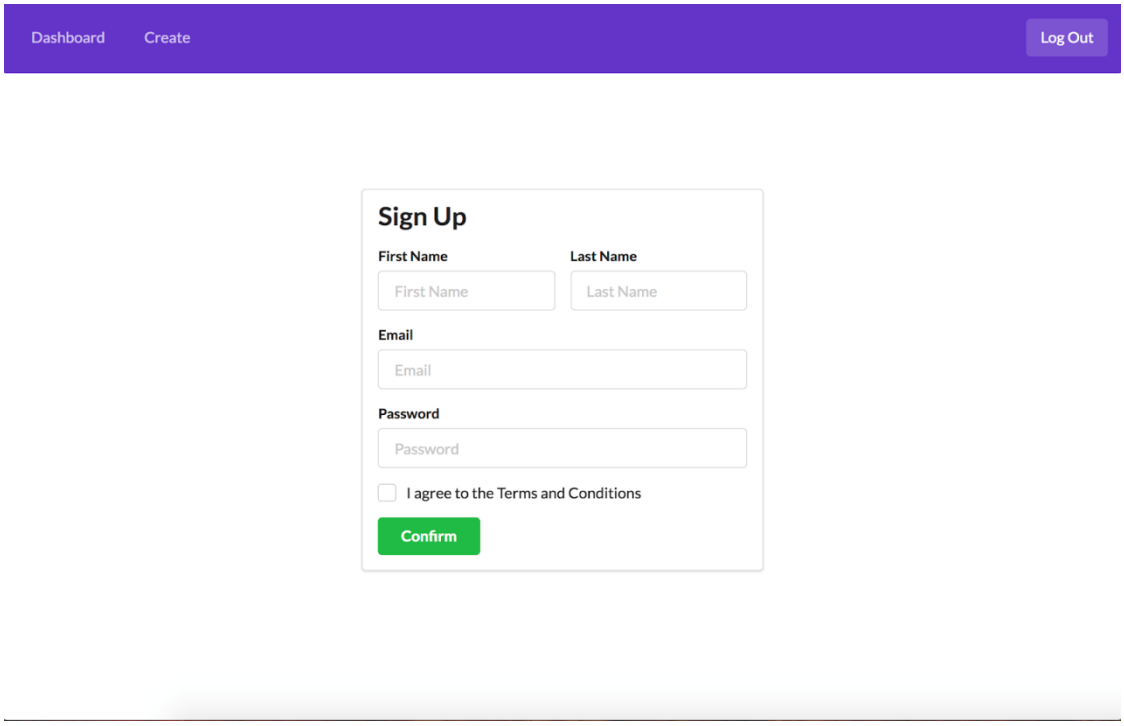


Рисунок 1.1 - Сторінка реєстрації

Після відправки даних система, у разі успішної реєстрації, повідомить користувача про створення нового облікового запису показом відповідного широкоформатного модального вікна, яке містить короткий коментар відносно процесу реєстрації та кнопку для продовження роботи. Модальне вікно зображене на рисунку 1.2.

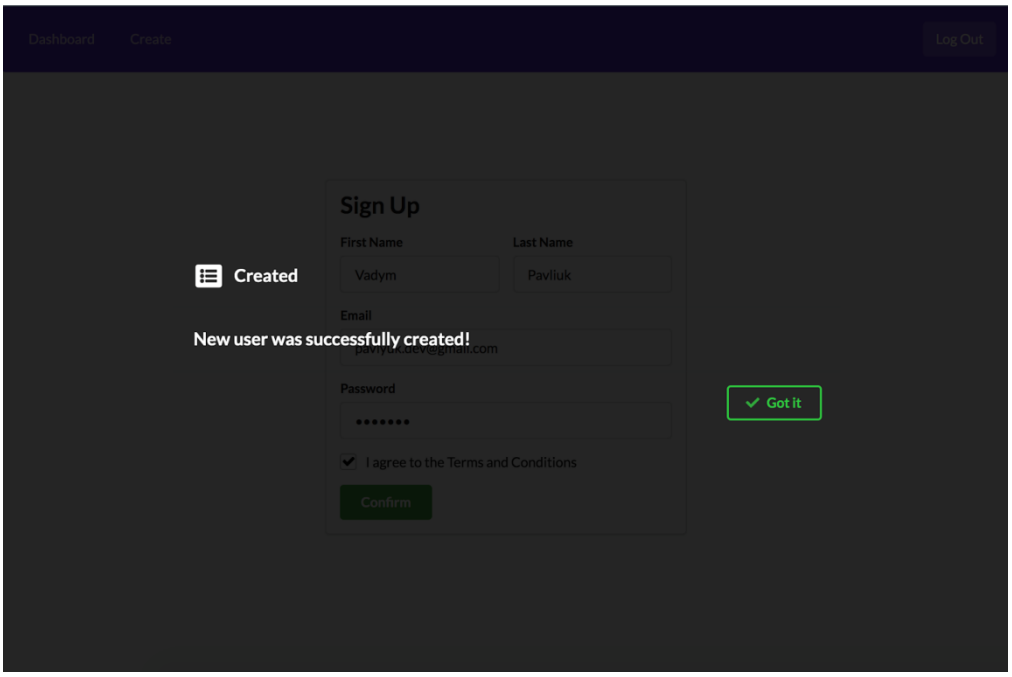


Рисунок 1.2 - Модальне вікно після успішної реєстрації

У разі виявлення помилок під час створення нового аккаунту система сповістить користувача відповідною підсказкою з інформацією, необхідною для виправлення проблеми.

1.2 Авторизація користувача

Для авторизації користувача необхідно перейти на сторінку авторизації за роутом ‘/sign-in’ та ввести контактну інформацію в поля введення. Для авторизації необхідно ввести електронну пошту та пароль. Після чого натиснути на кнопку відправлення даних. Сторінка авторизації зображена на рисунку 1.3. У разі виявлення помилок під час авторизації користувач отримає відповідну інформацію за домопогою модального вікна сповіщення.

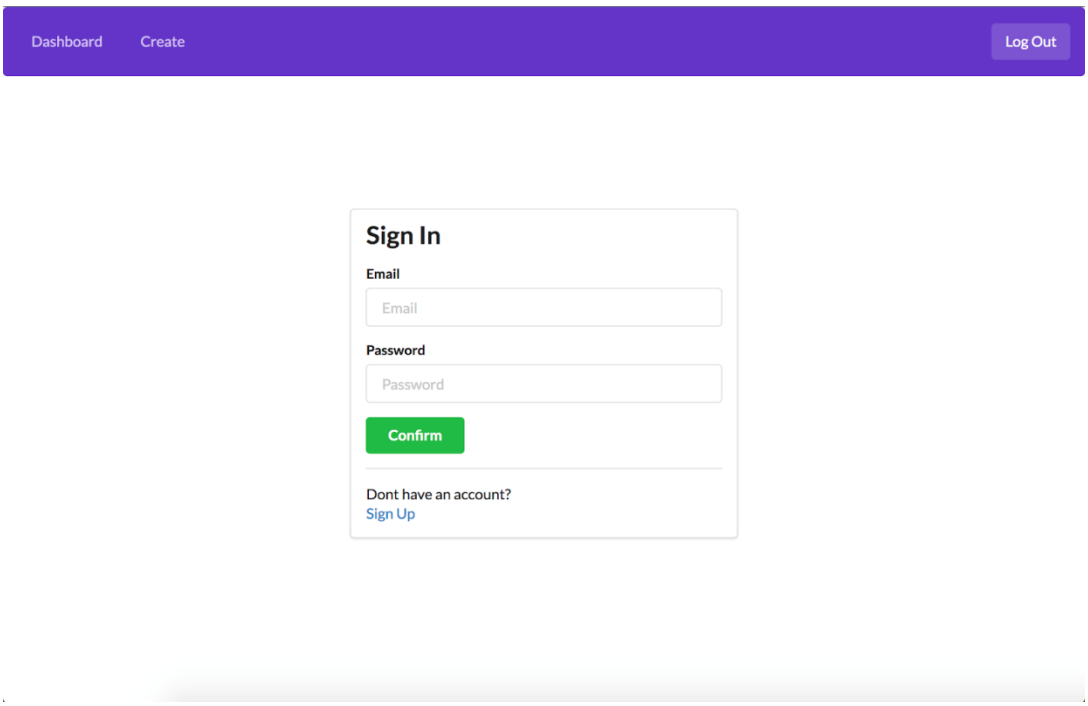


Рисунок 1.3 - Сторінка авторизації користувачів

1.3 Особистий кабінет

Після успішної авторизації користувач автоматично перенаправляється в особистий кабінет, де відображаються створені користувачем опитування, якщо такі існують, та опитування з якими користувач працював нещодавно. Сторінка особистого кабінету зображена на рисунку 1.4

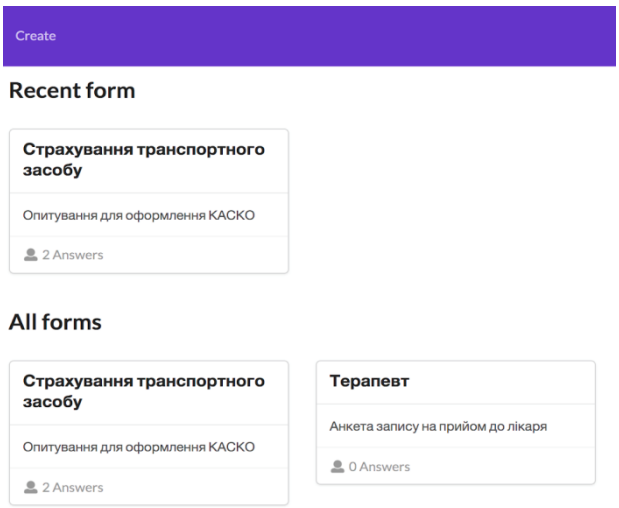


Рисунок 1.4 - Особистий кабінет

1.4 Перегляд опитування

Для того щоб, отримати інформацію про створене опитування необхідно натиснути на відповідну картку, яка складається з заголовка та короткого опису опитування. Після вибору картки користувач автоматично перенаправляється на сторінку опитування, яка зображена на рисунку 1.5. На даній сторінці у верхній частині знаходиться перемикач, який містить ідентифікатор відповідей на опитування та анкету. За замовчування активований режим перегляду відповідей на питання. Відповіді відображаються у вигляді таблиці, кожен рядок якої відповідає полю анкети опитування, а стовпець характеристиці цього поля. Також перед кожною таблицею відповідей знаходиться опис респодента: його контактна інформація та ідентифікатор, якщо відповідь надавав неавторизований користувач, він буде відображений як анонімний.

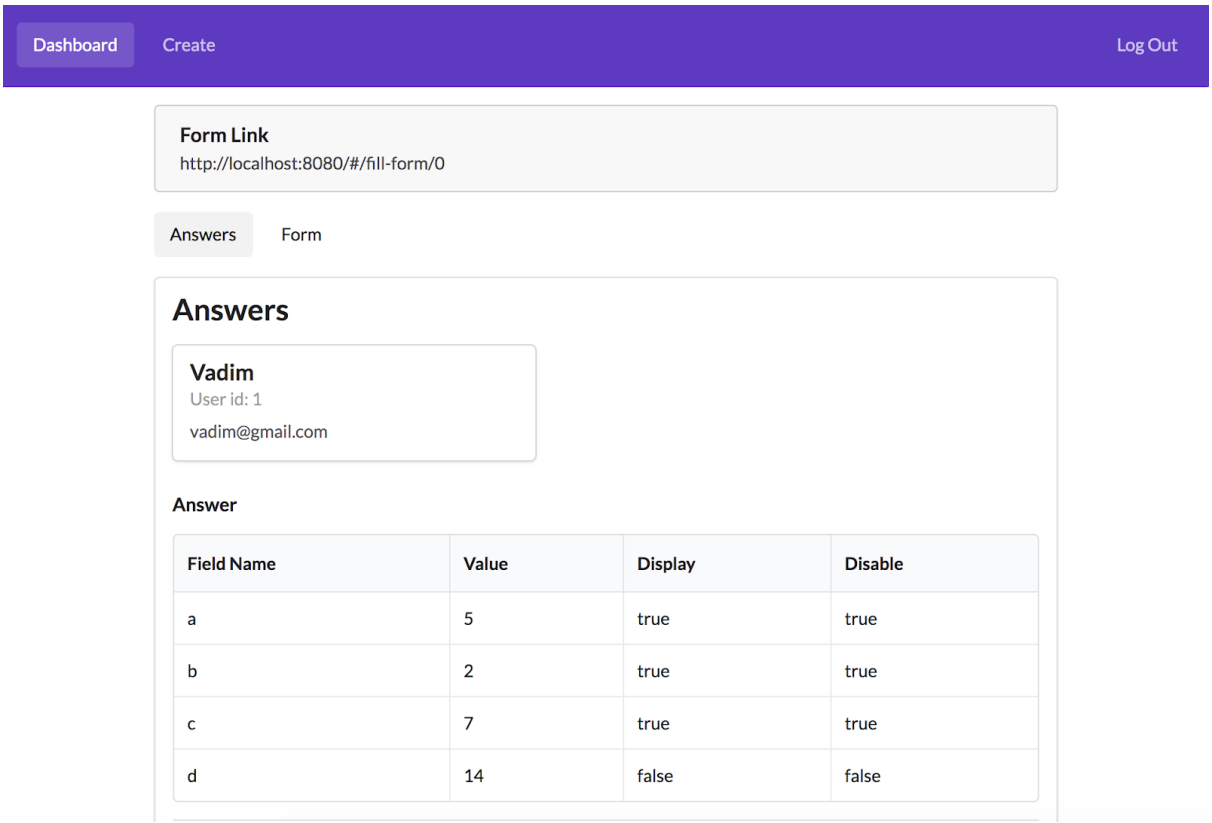


Рисунок 1.5 - Сторінка опитування

Для переключення в режим перегляду форми опитування необхідно активувати відповідний режим у верхній частині сторінки, після цього відбудеться відображення конструктора опитування. Переглянути даний конструктор можна на рисунку 1.6. Детальніше про даний конструктор у розділі створення опитування.

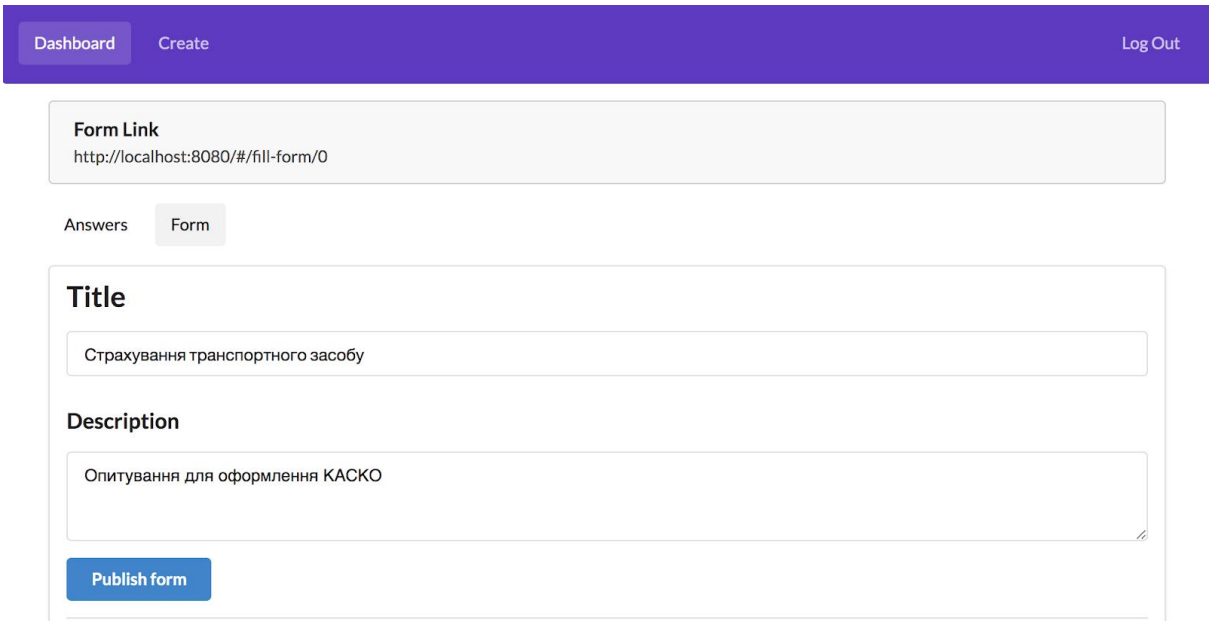


Рисунок 1.6 - Конструктор опитування

1.5 Створення опитування

Для створення опитування необхідно натиснути кнопку ‘створити’ у верхній фіксованій частині додатку. Після цього відбудеться перенаправлення користувача на сторінку створення опитування. На цій сторінці містить конструктор форм, ідентичний конструктору, що на сторінці перегляду анкети. Зображення даної сторінки та конструктора можна побачити на рисунку 1.7

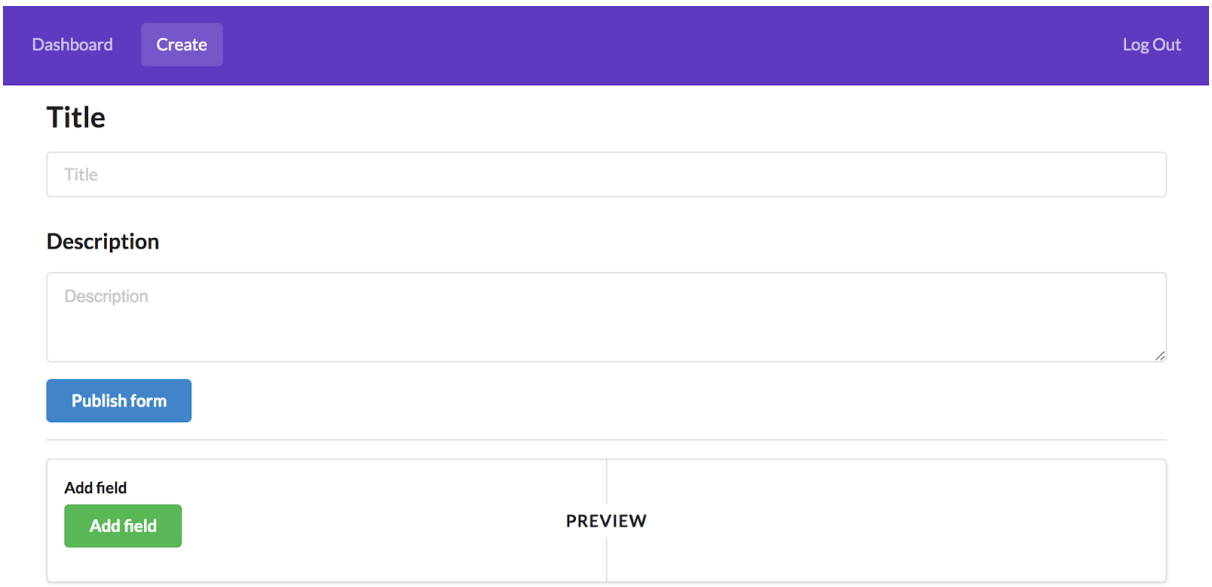


Рисунок 1.7 - Сторінка створення опитування

Користувач спочатку повинен ввести назву та опис опитування у відповідні поля, після чого натиснути кнопку додання нового поля. Після цього згенерується автоматичне поле з параметрами за замовчуванням.

До параметрів поля відносяться:

- назва поля;
- тип поля;
- заголовок поля;
- параметри стану поля;
  - значення;
  - відображення;
  - активність.

Кожен з цих параметрів представленим відповідним полем вводу в конструкторі, а саме поле відображається в інтерактивному емуляторі створюваної форми справа. Зображення конструктора з добавленим полем можна переглянути на рисунку 1.8

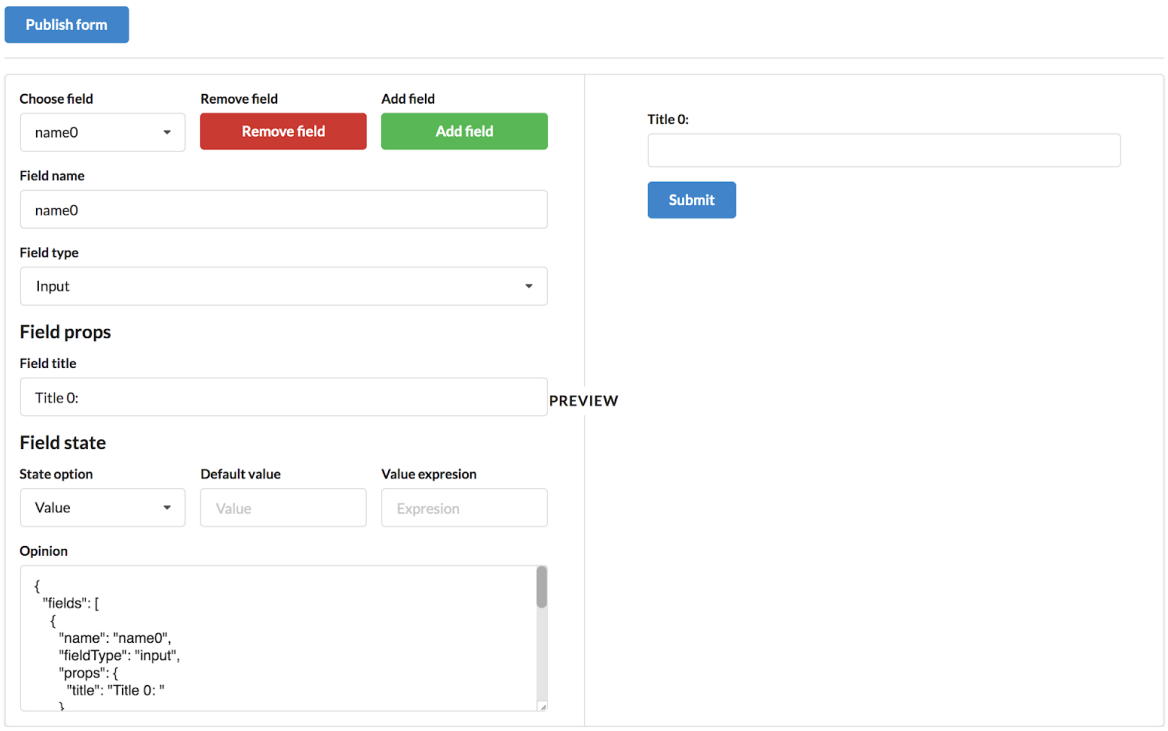


Рисунок 1.8 - Конструктор опитування з доданим полем

Шляхом вибранням відповідної опції можна вказати необхідний тип поля. Доступні типи полів зображені на рисунку 1.9

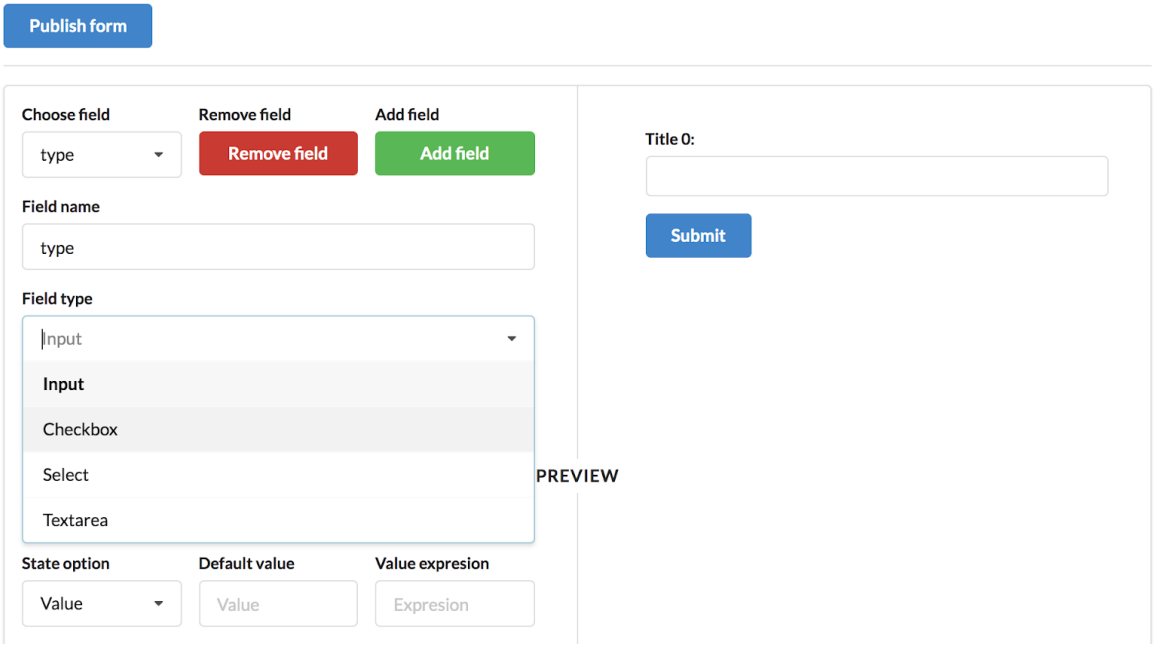


Рисунок 1.9 - Вибір типу поля

Після вибору типу поле одразу змінює свій вигляд, що в той же момент можна побачити в емуляторі анкети. Зображення емулятора після вибору типу поля знаходиться на рисунку 1.10

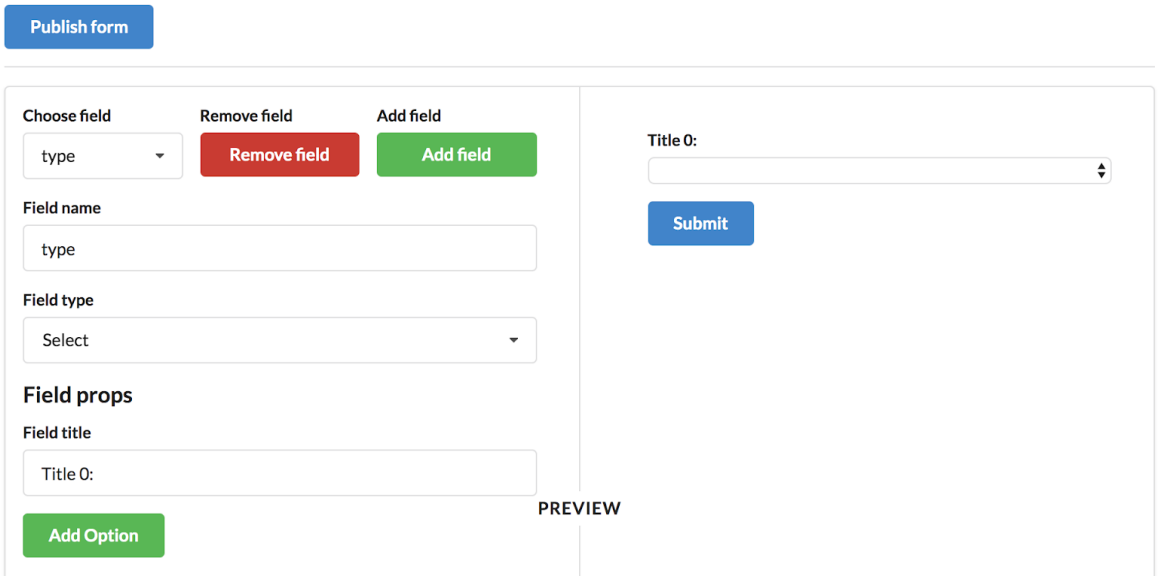


Рисунок 1.10 - Емулятор форми, після вибору типу поля

При виборі у ролі типу поля випадаючого списку з’являється кнопка добалення опцій. Після натиснення цієї кнопки з’являються поля для додання даних щодо нової опції випадаючого списку. Ці поля включають поле вводу значення опції та поле вводу назви опції, навпроти опції знаходиться кнопка видалення опції. Поля добавлення опцій зображені на рисунку 1.11

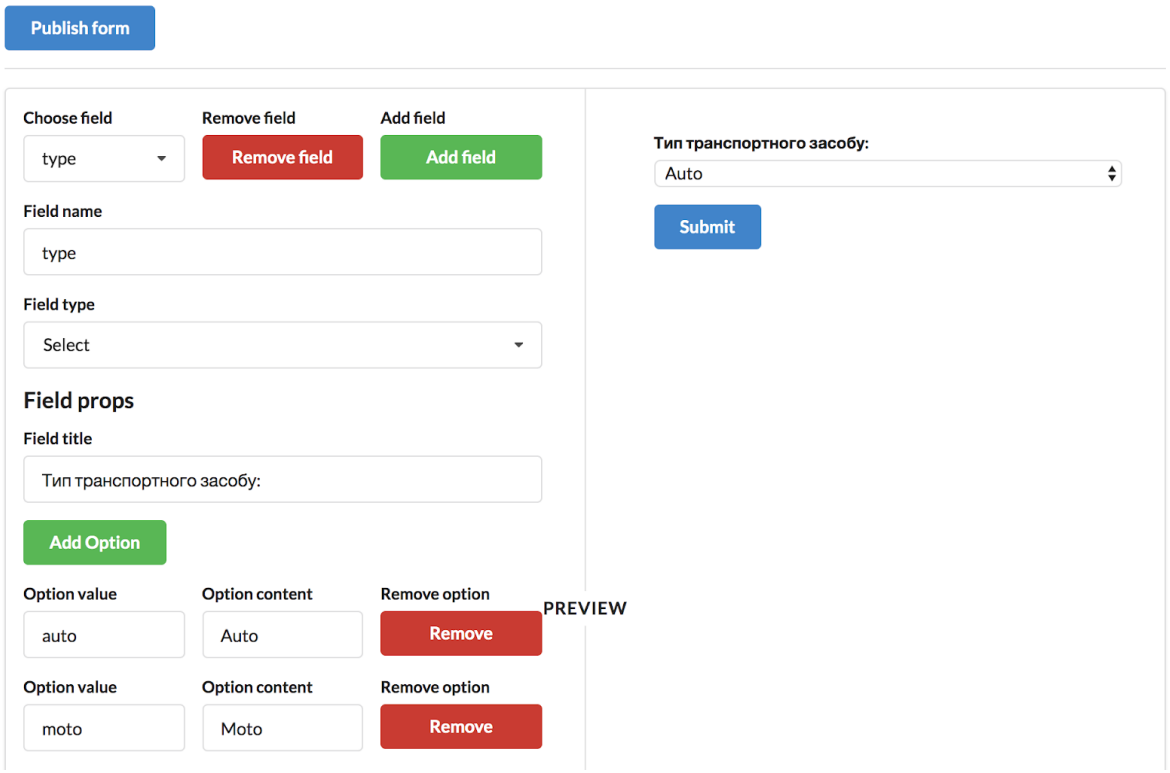


Рисунок 11 - Поля добавлення опцій для випадаючого списку

Після заповнення значення опцій вони в цей же ж момент з’являються в інтерактивному емуляторі, де можна протестувати їх роботу. Роботу емулятора після добавлення опцій можна переглянути на рисунку 1.12

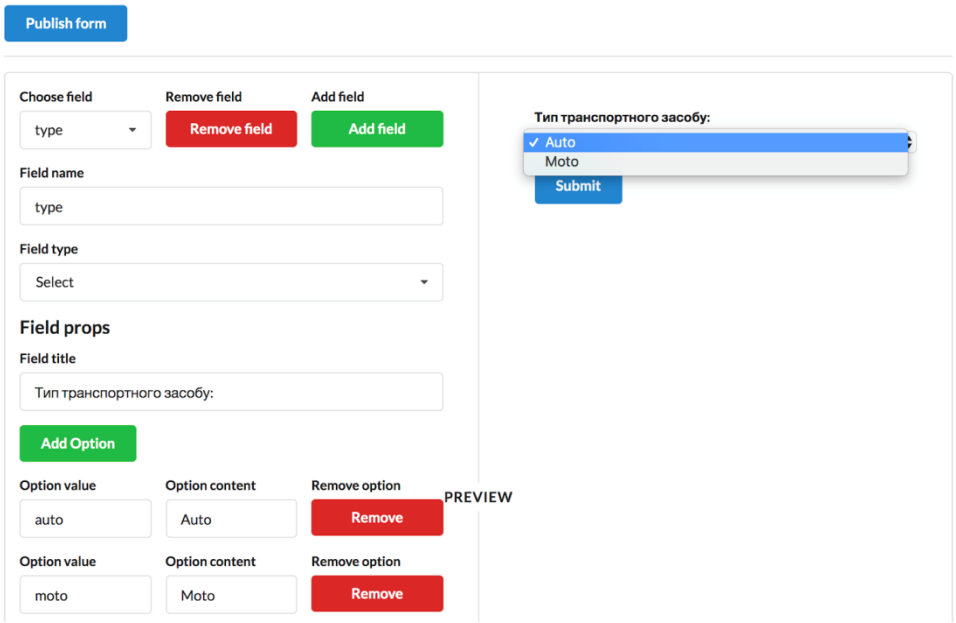


Рисунок 1.12 - Стан емулятора після добавлення опцій в поле

1.5.1 Налаштування залежностей між полями

Кожне поле містить блок керування внутрішнім станом поля. За допомогою цього стану можна налаштовувати зв`язки між полями, а саме автоматичну зміну значення, відображення, активності поля. Для демонстрації процесу налаштування зв’язків створимо два текстових поля для введеня інформації щодо моделі авто та мото.

Поточний стан анкети можна переглянути на рисунку 1.13

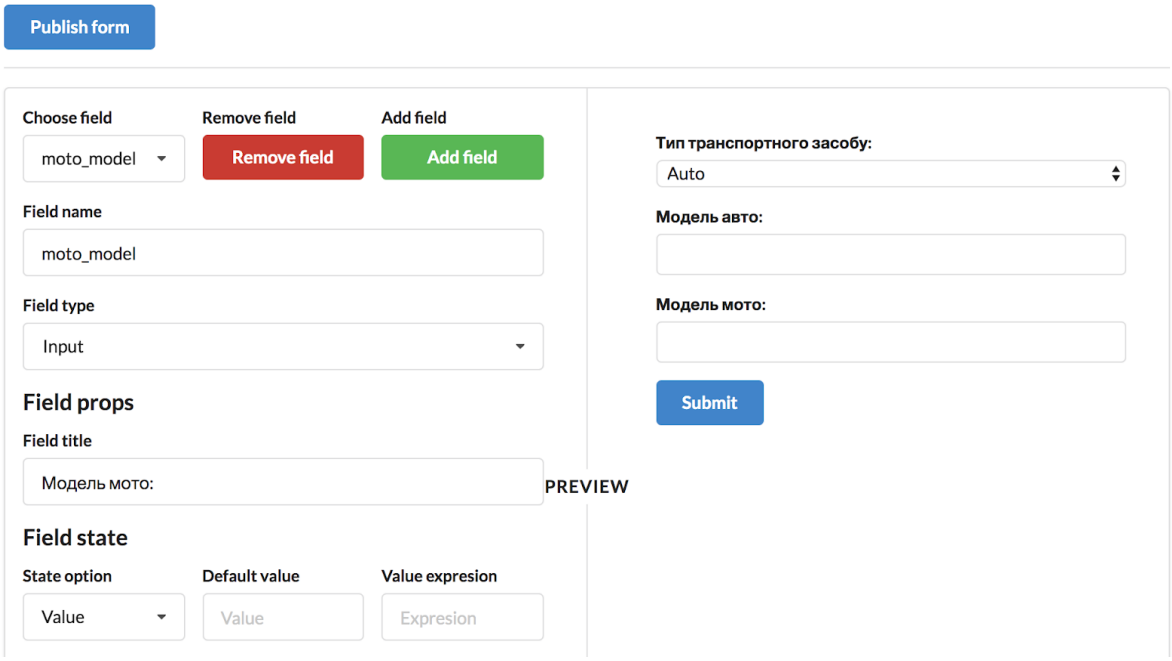


Рисунок 1.13 - Стан емулятора після добавлення полів вводу моделей

Тепер перейдемо до налаштування відображення доданих полів. Для того, щоб зробити правильне їх відображення необхідно додати зв’язок із полем типу транспортного засобу і залежно від його значення відображати або приховувати дані поля, які будуть залежними від батьківського. Для створення зв’язку необхідно перейти до вибору опції відображення в блоці налаштування стану поля. Опції стану поля зображені на рисунку 1.14

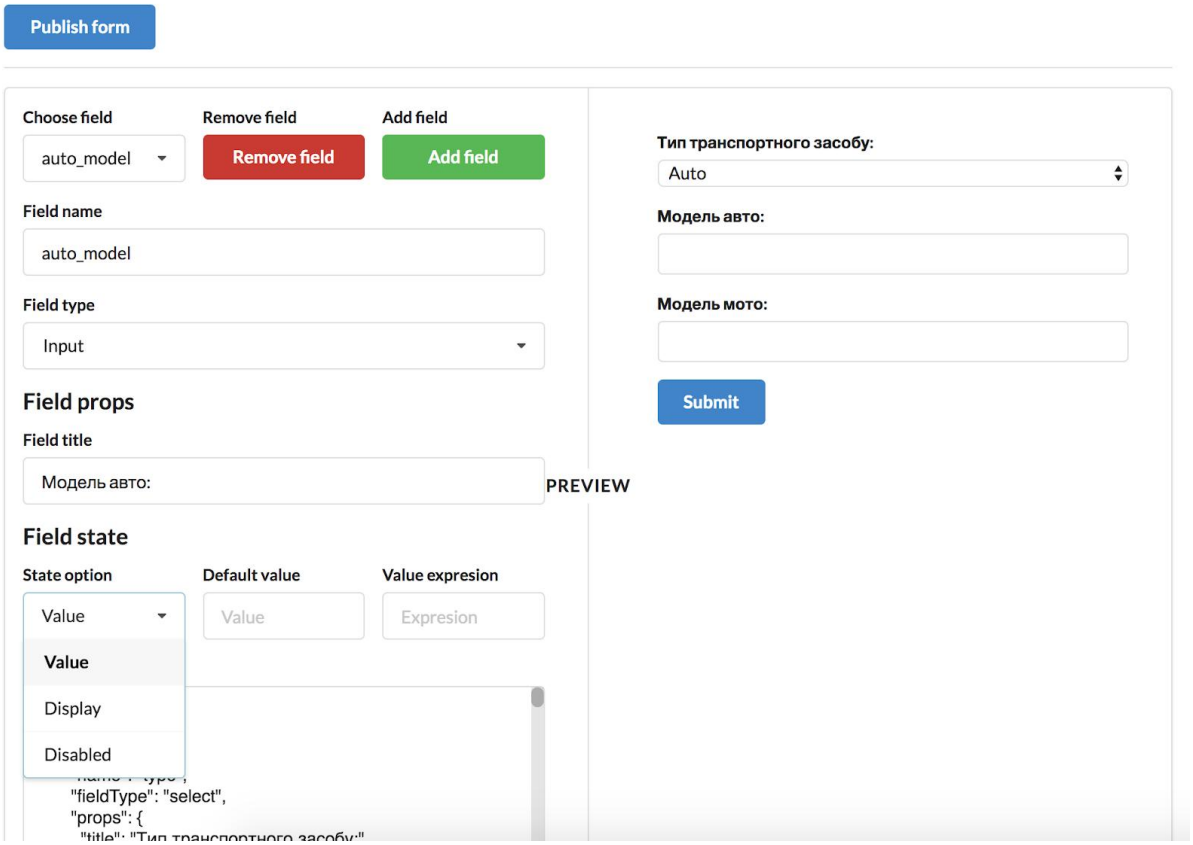


Рисунок 1.14 - Опції стану поля

Для реалізації нашого завдання необхідно вибрати опцію відображення та написати умову відображення, використовуючи значення поля типу транспортного засобу. Для поля ‘моделі авто’ умовою буде значення поля типу ‘авто’, для поля ‘моделі мото’ відповідно - ‘мото’. Для формалізації відповідної умови необхідно в поле виразу ввести логічну умову порівняння значення поля ‘типу’ з константою використовуючи операнд порівняння ‘==’. Приклад налаштування логічного зв’язку між полями наведений на рисунку 1.15

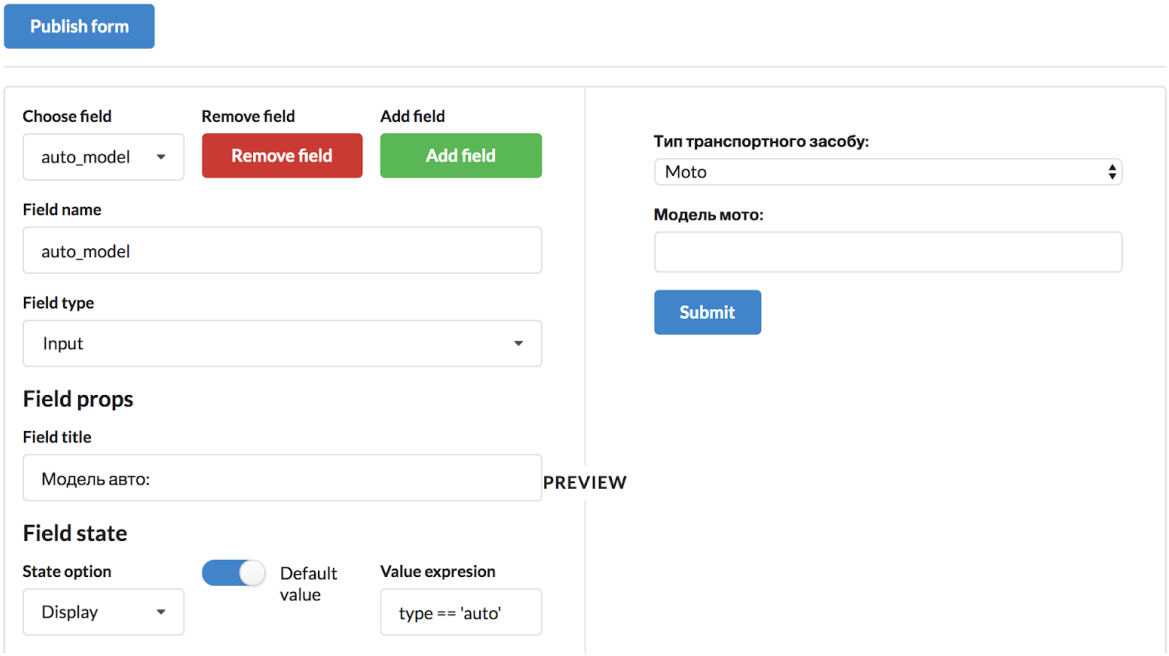


Рисунок 1.15 - Приклад налаштування логічного зв’язку між полями

Перейдемо до налаштування автозаповнення на базі залежностей між полями. Для прикладу розглянемо схематичну залежність нового поля ‘бажана страхова сума’ від початкової ціни транспортного засобу та його віку. Після добавлення нових полів, перейдемо до налаштування опції стану - значення поля. Вигляд форми після добавлення нових полів та встановлення значення за замовчуванням буде виглядати як на рисунку 1.16

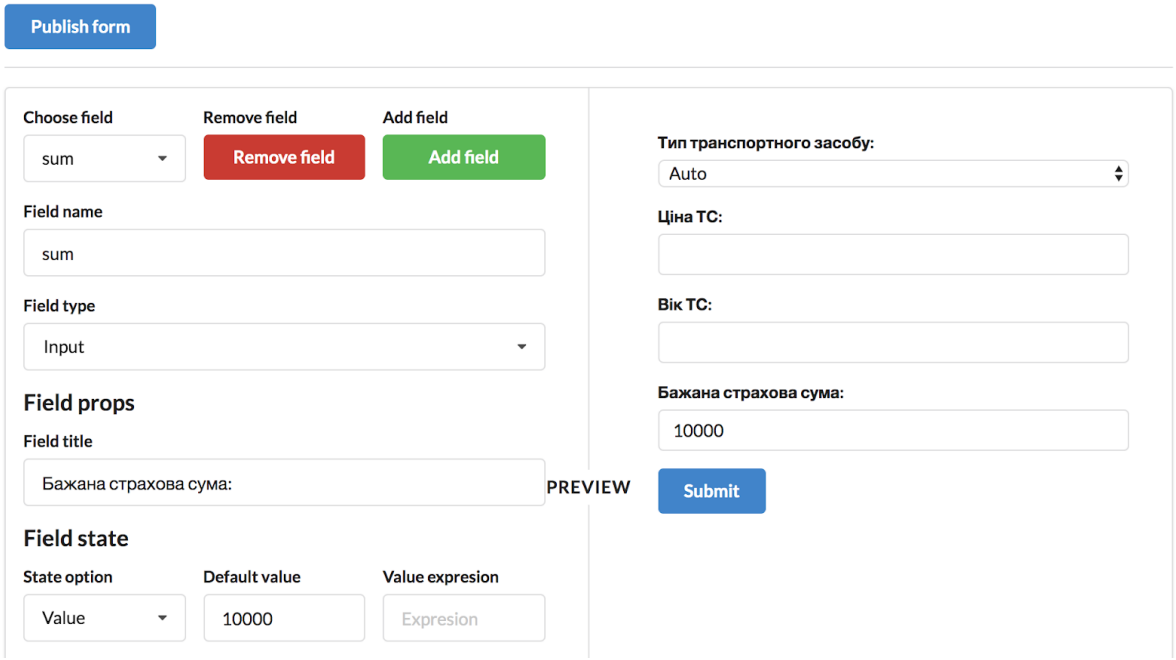


Рисунок 1.16 - Анкета після додання полів обчислення страхової суми

Для того щоб замінити значення за замовчуванням на автоматичний підрахунок значення залежно від оновлення батьківських полів введемо математичний вираз, використовуючи загальноприйняту алгебру. Значення бажаної страхової суми буде формалізоване у вигляді виразу ‘price * ((10 - age) / age)’. Змінивши значення будь-якого із батьківських полів отримаємо автоматичне оновлення поля страхової суми. Залежність між полями на базі математичного виразу зображена на рисунку 1.17

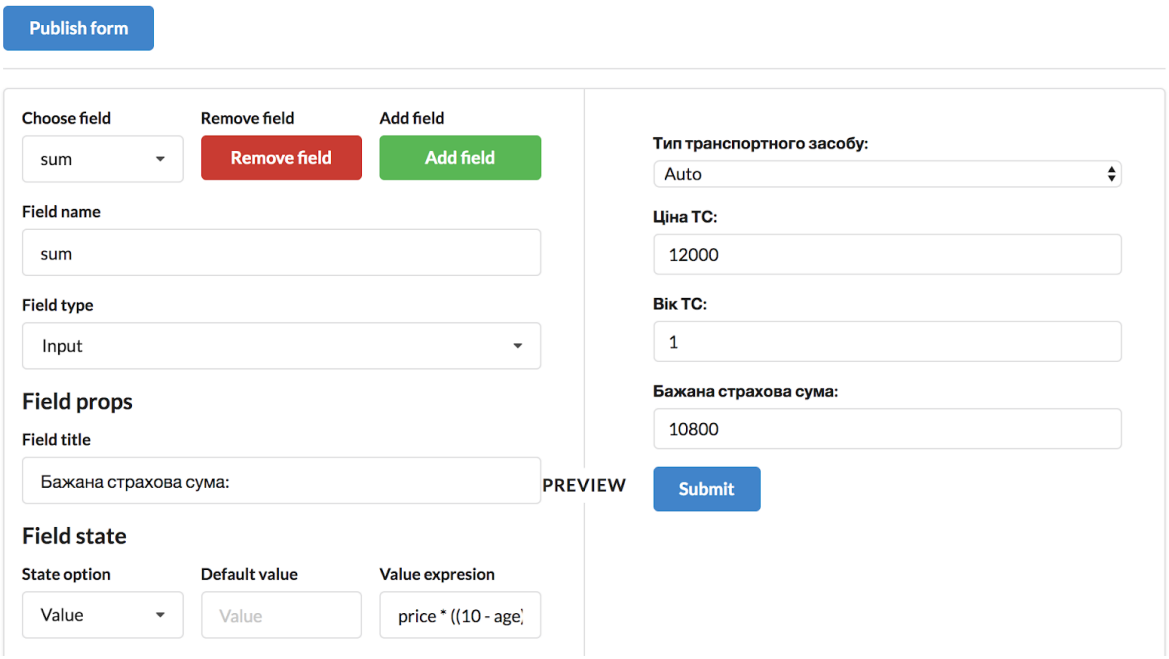


Рисунок 1.17 - Залежність між полями на базі математичного виразу

1.5.2 Збереження створеного опитування

Перед відправкою опитування необхідно ввести його заголовок та короткий опис у відповідні поля, що знаходяться у верхній частині сторінки, перед конструктором опитування. Поля заповнення опису опитування зображені на рисунку 1.18

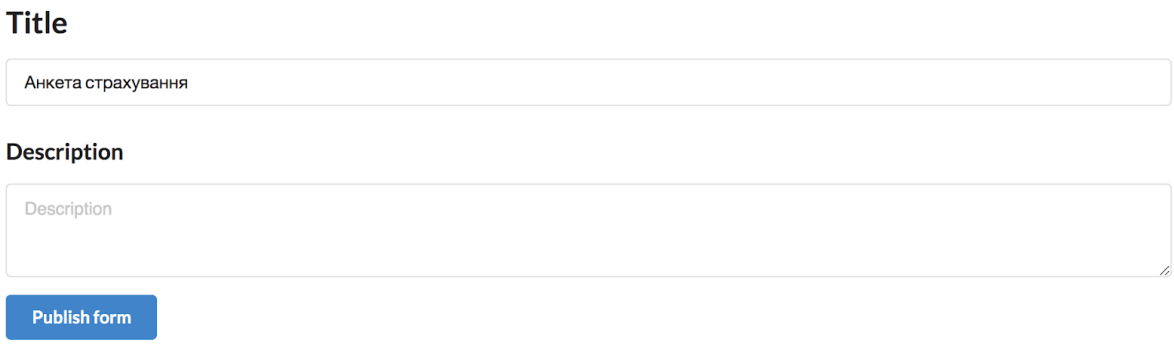


Рисунок 1.18 - Поля заповнення опису опитування

Для збереження опитування необхідно натиснути кнопку публікації, після чого з’явиться модальне вікно з повідомленням про успішне створення опитування. На модальному вікні розташована кнопка продовження, після натискання на неї користувач перенаправиться в особистий кабінет, де повинне з’явитися щойно створене опитування. Особистий кабінет з створеним опитуванням зображений на рисунку 1.19

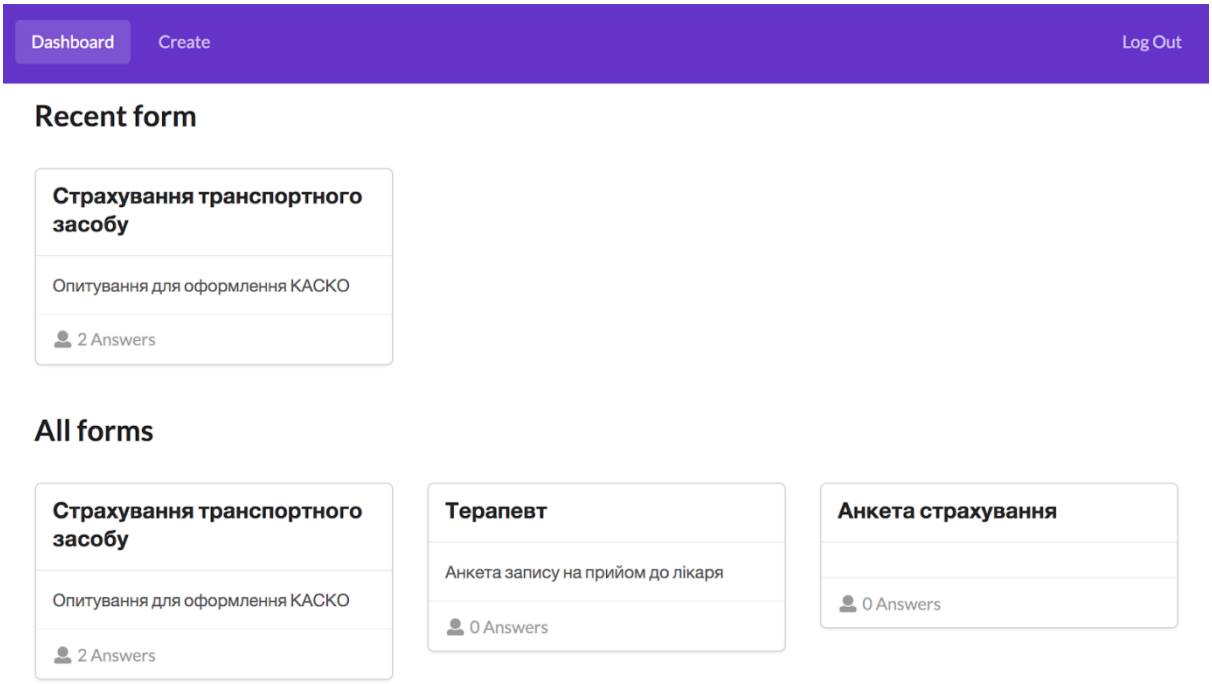


Рисунок 1.19 - Особистий кабінет з створеним опитуванням

1.6 Проходження опитування

Для того, щоб пройти опитування респоденту необхідно отримати посилання на форму опитування. Дане посилання знаходиться на сторінці перегляду опитування у верхній її частині над перемикачем режиму сторінки. Із зображення посилання на опитування можна ознайомитися на рисунку 1.20

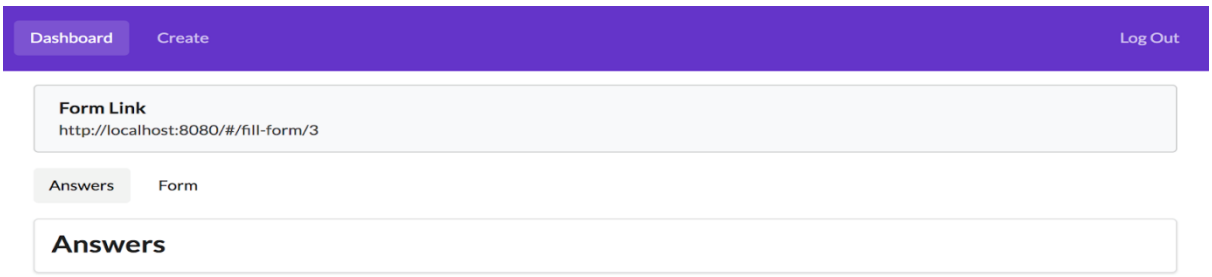


Рисунок 1.20 - Зображення посилання на опитування

Скопіювавши посилання, його необхідно надіслати респоденту будь-яким зручним способом. Після чого, користувач, перейшовши за посиланням, отримає анкету у вигляді динамічної форми ідентичну тій, що була в емуляторі конструктора під час створення анкети. Приклад заповнення анкети продемонстрований на рисунку 1.21

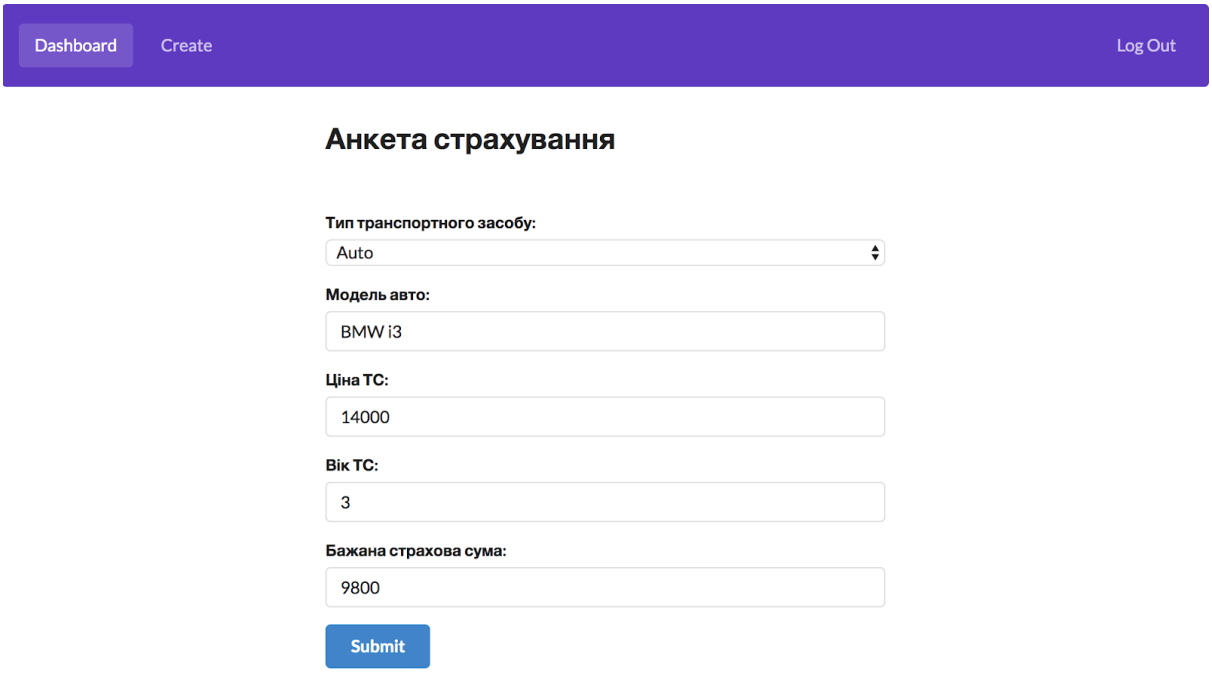


Рисунок 1.21 - Приклад заповнення анкети

Після натиснення кнопки відправити користувач сповіщається про успішне заповнення анкети показов відповідного модального вікна. Модальне вікно, яке

повідомляє про успішну відправку відповідей на опитування зображено на рисунку 1.22

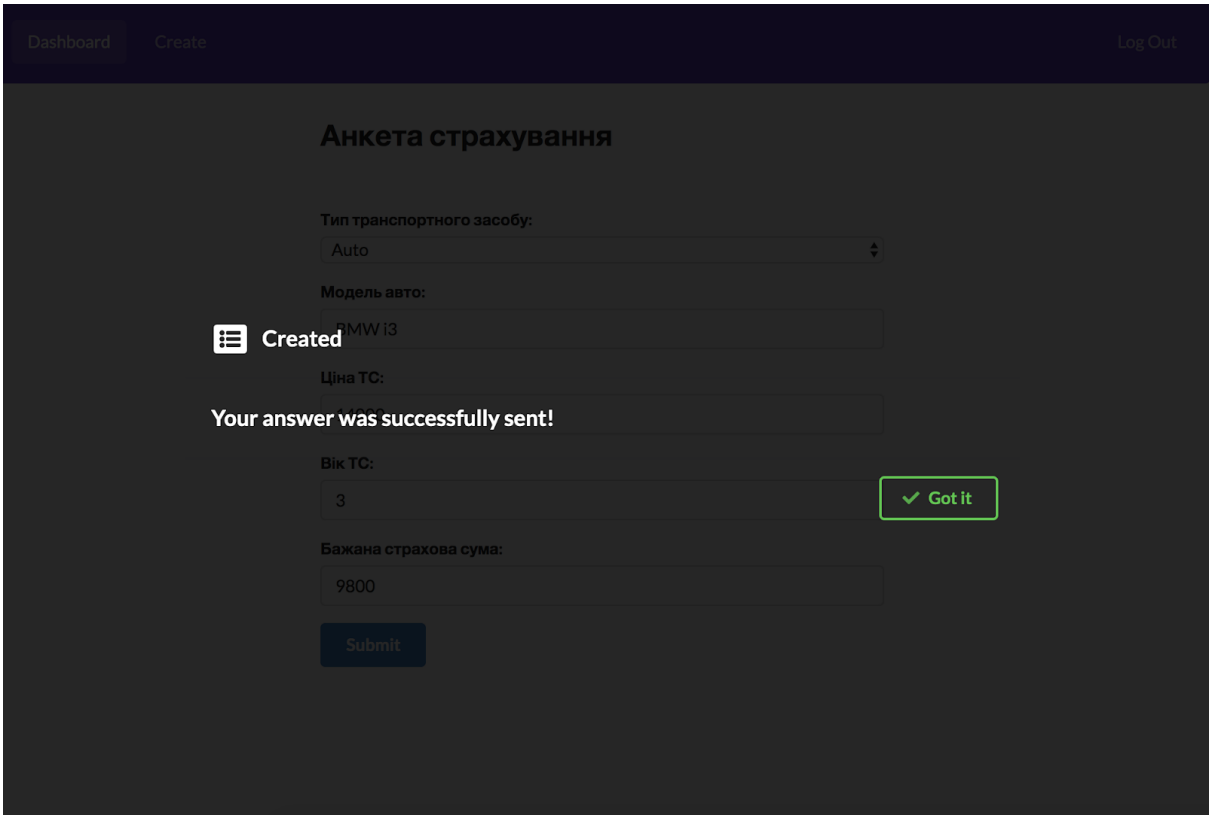


Рисунок 1.22 - Приклад заповнення анкети

1.7 Перегляд результатів опитування

Після відправки респодентом відповідей, вони одразу стають доступні власнику форми на сторінці перегляду опитування. Відповіді розташовуються блоками, в кожному з яких є інформація про користувача, який надавав дані відповіді, якщо він був авторизований на той момент, та самі відповіді у вигляді таблиці із усіма полями та станами цих полів.

Приклад результатів опитування зображений на рисунку 1.23

ї

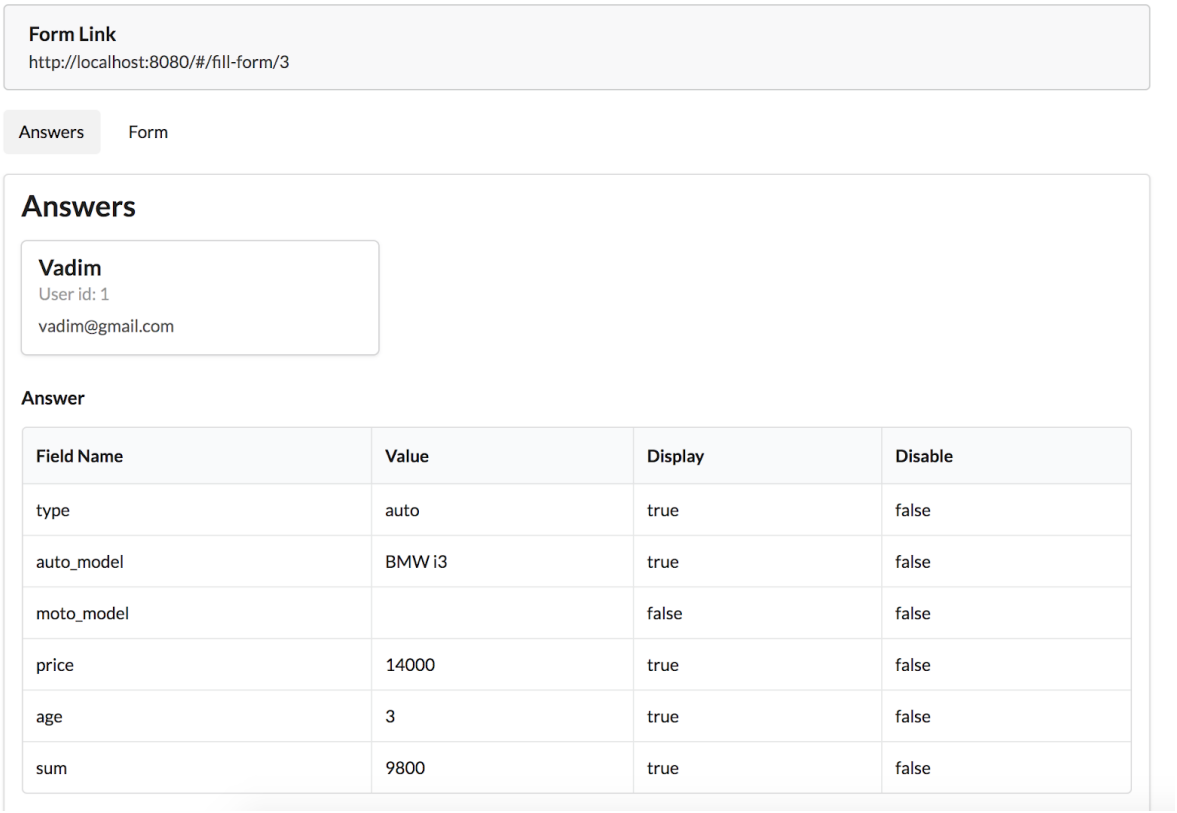


Рисунок 1.23 - Приклад результатів опитування

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ О.А. Павлов

“___” _____ 2019 р.

Комплекс задач з автоматизації варіативних опитувань

Графічні матеріали

КПІ.ІП-5115.045440.06.99

“ПОГОДЖЕНО”

Керівник проекту:

_____О.В. Ковтунець

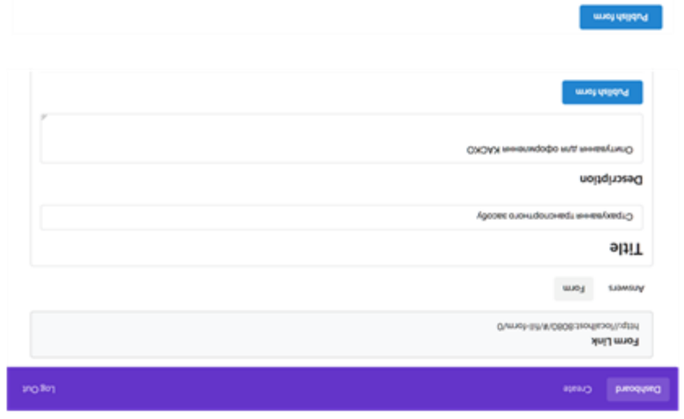
Нормоконтроль:

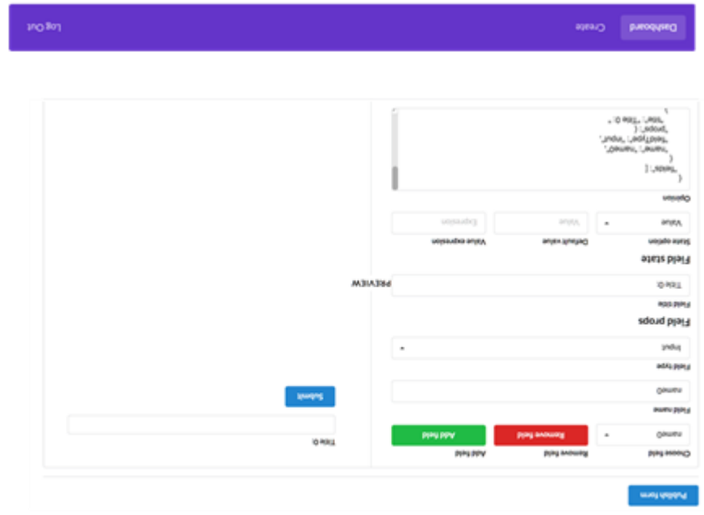
_____ К.І. Ліщук

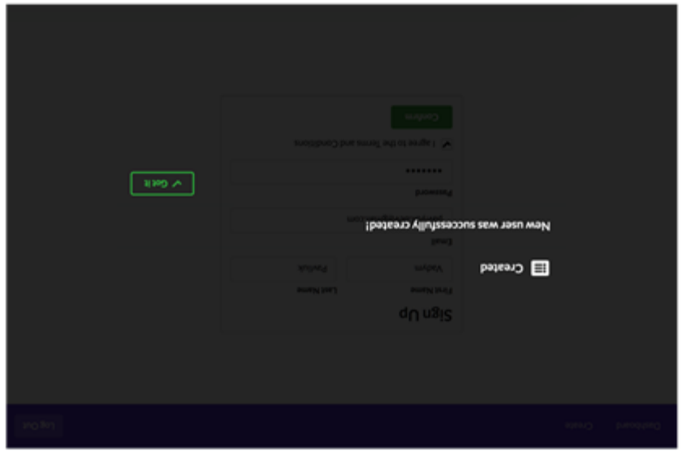
Виконавець:

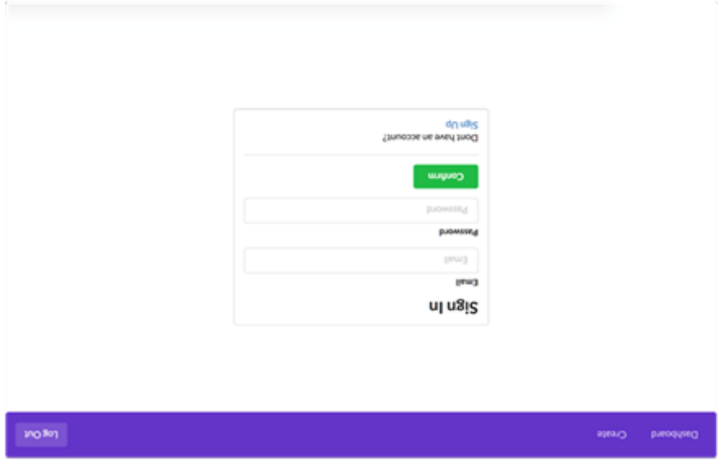
_____В.Р. Павлюк

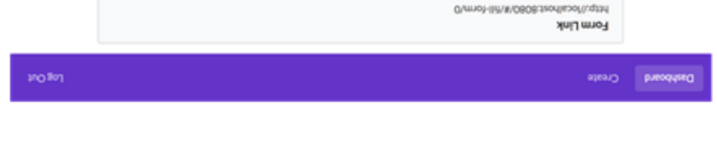
Київ – 2019 року











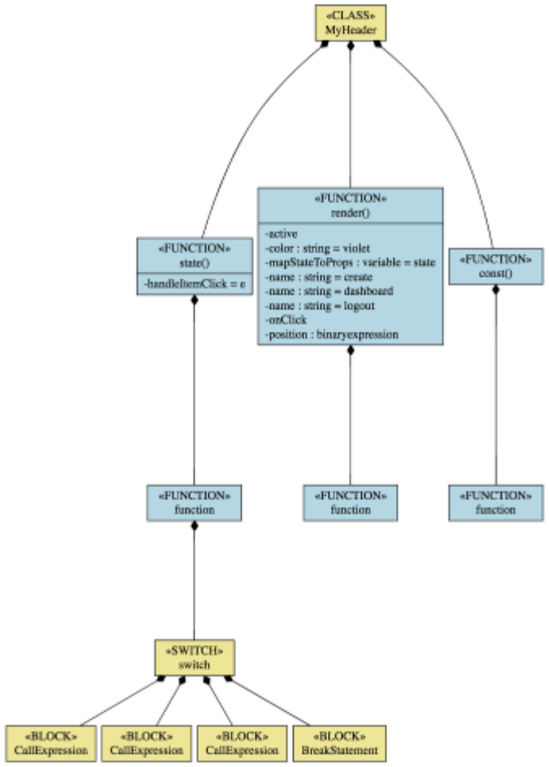
Зм.	Арк.	№ докумнта	Підпис	Дата	
Розробив		Гавлюк В.Р.			
Перевірив		Ковтуньєв О.В			
Т. контр.					
Н. контр.		Лицук К.І.			
Затвердив		Ковтуньєв О.В			

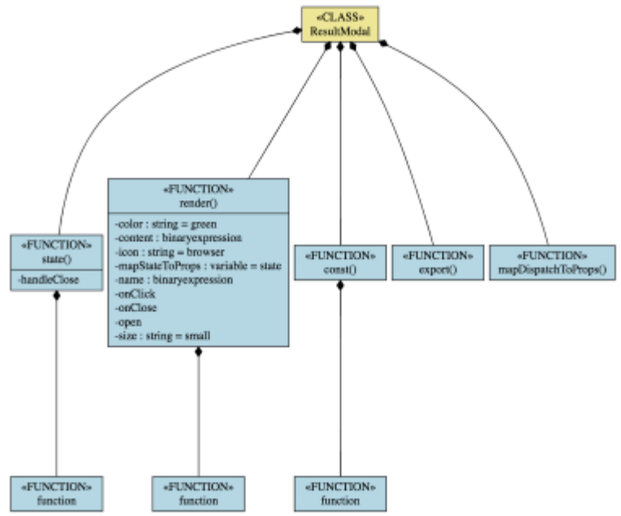
КПІ.ІП-5115.045440.06.99.КЕ

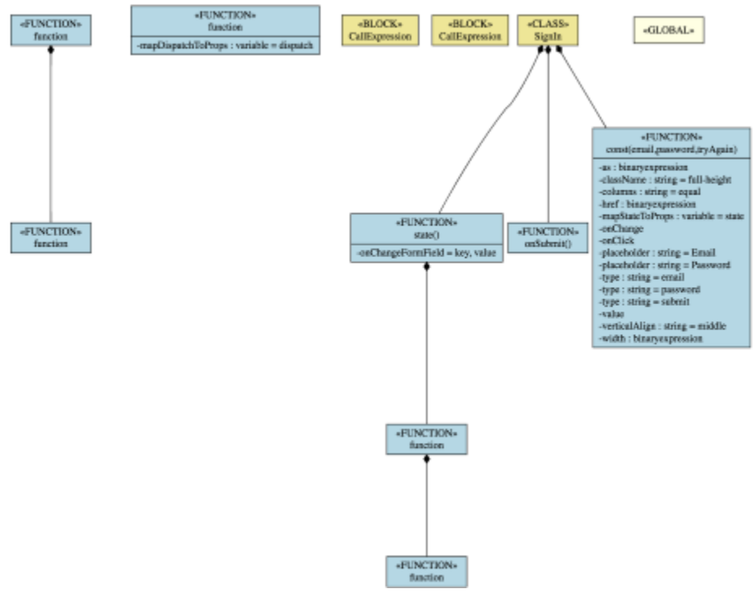
Кресленик випускної екранних форм

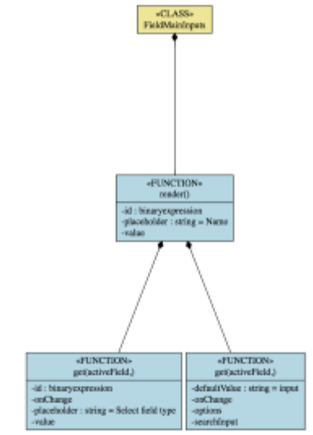
Літера	Маса	Масштаб
Аркуш	Аркушів	

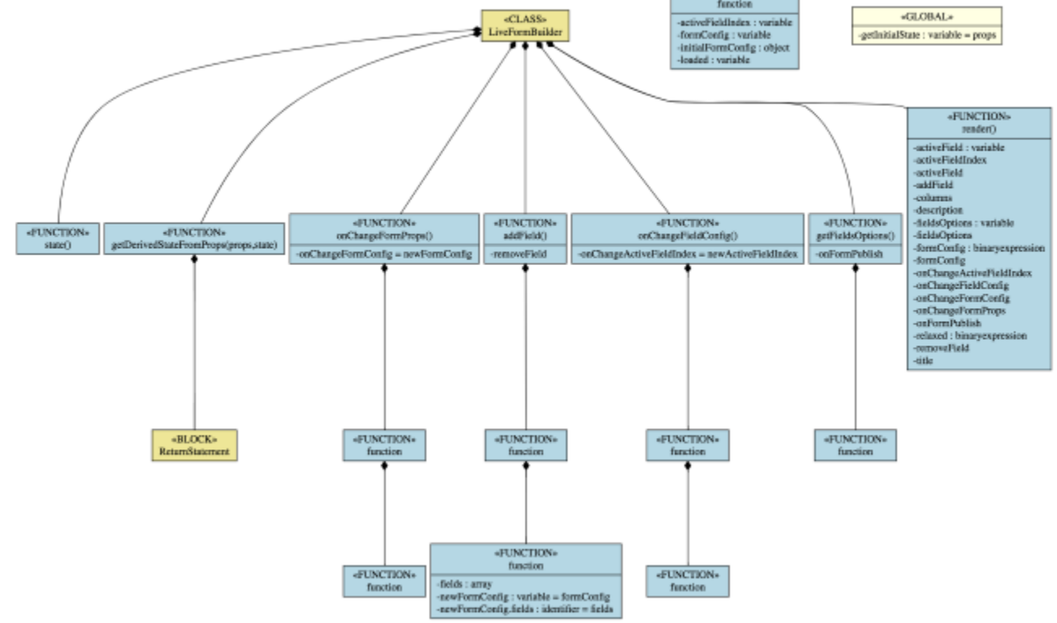
КПІ ім.Ігоря Сікорського
Кафедра АСОІУ
гр. ІП-51

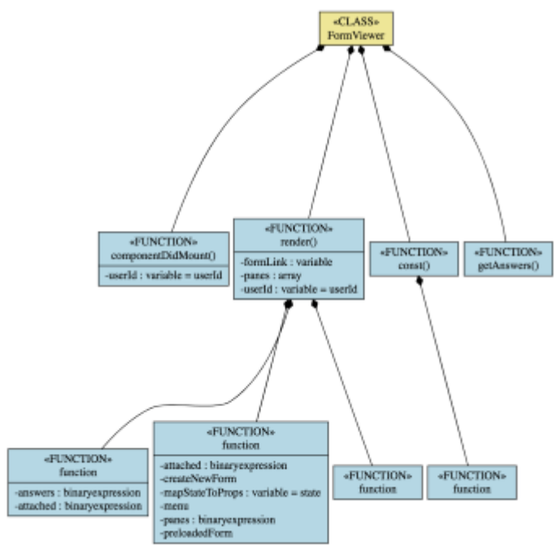


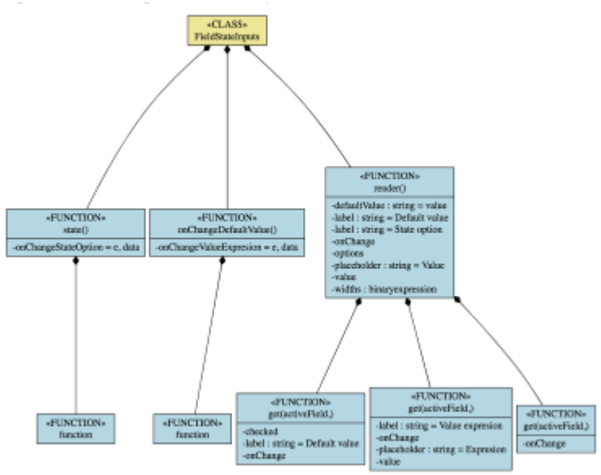






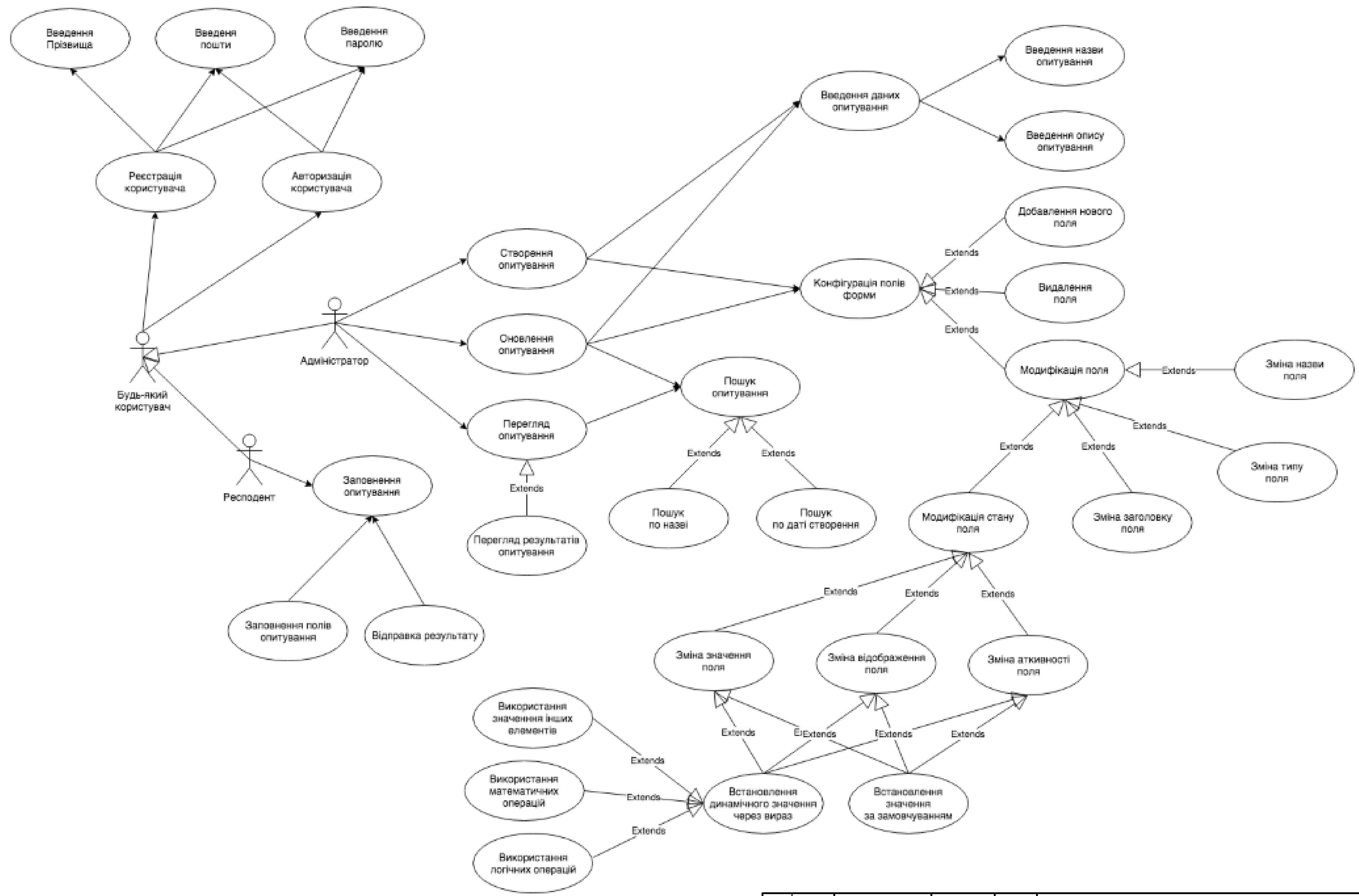






					КПІ.ІП-5115.045440.06.99.СС	
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна класів програмного забезпечення	Літера / Маса / Масштаб
Розробив		Павлюк В.Р.				
Перевірив		Ковтунець О.В.				Аркуш / Аркушів
Т. кон.						
Н. кон.		Ліщук К.І.			Комплекс задач з автоматизації варіативних опитувань	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-51
Затвердив		Ковтунець О.В.				

КПІ.ІП-5115.045440.06.99 СС



					КПІ.ІП-5115.045440.06.99 СС
Зм.	Арк.	№ документа	Підпис	Дата	
Розробив		Павлюк В.Р.			Схема структурна варіантів використання
Перевірив		Ковтунець О.В			Літера / Маса / Масштаб
Т. кон.					Аркуш / Аркушів
Н. кон.		Ліщук К.І.			Комплекс задач з автоматизації варіативних опитувань
Затвердив		Ковтунець О.В.			КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-51